An Adaptive Ensemble of On-line Extreme Learning Machines with Variable Forgetting Factor for Dynamic System Prediction

Symone G. Soares*, Rui Araújo

Institute for Systems and Robotics, and Department of Electrical and Computer Engineering, University of Coimbra, Pólo II, PT-3030-290 - Coimbra, Portugal.

Abstract

A demand for predictive models for on-line estimation of variables is increasing in industry. As industrial processes are time-varying, on-line learning algorithms should be adaptive to capture process changes. On-line ensemble methods have been shown to provide better generalization performance than single models in changing environments. However, most on-line ensembles do not include and exclude models during on-line operation. As a result, the ensembles have limited adaptation capability. Moreover, a higher performance can be obtained by combining a selected set of most relevant models of the ensemble for the current situation, rather than combining all the models. This paper proposes a new on-line learning ensemble of regressor models using an ordered aggregation (OA) technique which is able to provide on-line predictions of variables in changing environments. OA dynamically selects an optimal size and composition of a subset of models based on the minimization of the ensemble error on the newest sample. The proposed strategy overcomes the problem of defining the optimal ensemble size, and in most cases it obtains better performance than aggregating all the models. Models are added or removed for assuring adaptation of the ensemble in changing environments. Furthermore, this paper proposes and integrates a new on-line Extreme Learning Machine (ELM) neural network model with variable forgetting factor (FF) using the directional FF method which shows superior performance in industrial applications when compared to the well-known On-line Sequential ELM (OS-ELM) algorithm. Experiments are reported to demonstrate the performance and effectiveness of the proposed methods.

Keywords: On-line Extreme Learning Machines; On-line ensemble; Ordered aggregation; Variable forgetting factor;

1. Introduction

Artificial Neural Networks (ANNs) have been widely investigated to solve problems in time series prediction [1], pattern recognition [2], and industries in the recent decades [3]. However, ANNs are rarely used in the industry or in real-time applications, because the networks need large training time and large training data to perform well. The reasons are that the network structure and training parameters need to be carefully chosen; and the learning phase may take a long time (e.g. back propagation algorithm [4]). Recently, the Extreme Learning Machine (ELM) has been attracting attention among the scientific community [5]. ELM is a single hidden layer feedforward ANN [6]. The input weights and biases are chosen randomly, and the output weights are determined analytically by the Least Squares (LS) method, allowing significant training time reduction when compared to other models; and demonstrating ability to deal with non-linear problems and good generalization performance [7].

Many practical systems, such as industrial plants, exhibit time-varying behavior, being very difficult for the ELM models to react to the changes. Recently, ELM models for dynamic environments have been proposed. The most popular is the on-line sequential ELM (OS-ELM) [6]. It can learn samples on-line using concepts of the Recursive Least Squares (RLS) algorithm [8]. In [9], it is proposed an On-line Sequential Reduced Kernel ELM that is incrementally updated based on the new samples' confidence estimation. In [10], an OS-ELM algorithm with Kernels (OS-ELMK), which replaces the hidden layer mapping in ELM by the kernel function mapping in the support vector machine, is proposed. OS-ELMK proposes a decremental mechanism to remove the oldest trained samples from the model when the number of trained samples exceeds a threshold. In [11], a forgetting factor (FF) is introduced to the OS-ELM algorithm. When the FF value is close to 1 more contribution is given to the old samples, and when the FF value is close to 0 more importance is given to the recent samples. Since a fixed value for the FF may not be sufficient to track all the system dynamics, a variable FF for the OS-ELM is proposed in [12]. The FF is adapted using a gradient (derivative) descent method, derived from a cost function of the RLS. This method depends on the appropriate step size and takes too many iterations to converge to the appropriate FF value. RLS with FF discounts continuously the old data even when the new data does not carry sufficient information, producing a phenomenon known as *windup* [13]. As a consequence, values of the information matrix will tend to zero and the model gain will

*Corresponding author

Email addresses: symonesoares@isr.uc.pt (Symone G. Soares), rui@isr.uc.pt (Rui Araújo)

tend to be unbounded, so that the model becomes sensitive to noises. The directional FF (DFF) method can overcome this effect [14, 15]. It considers that the data has directions, and the old samples are forgotten only in some specific directions.

Although, ELM has shown good performance in real-world applications, recent studies have been shown that a better generalization performance can be achieved by combining a set of ELM models rather than using a single ELM model [16, 17]. This technique is known as ensemble learning method. On-line ensemble algorithms have been shown to be able to improve the prediction performance in time-varying systems [18, 19]. They can perform a subset of the following strategies: models' weights adaptation; models' parameters adaptation; and/or new model inclusion over time. Ensembles are classified as sample-based or batch-based if they are adapted when a sample or a batch of samples is available, respectively. Most ensembles are batch-based, e.g. the Learn^{++}.NSE [20, 21]. Batch-based ensembles usually require a long time to wait for a batch, and when a batch becomes available such data may not reflect the current state of the process. Sample-based ensembles offer faster adaptivity in changing environments and good performance in applications where variables are measured at low sampling rates.

Examples of on-line sample-based ensemble algorithms are the On-line Bagging (OB) [22] and the ensemble of OS-ELM (EOS-ELM) [17]. They have few mechanisms to track changing systems, since only the models' parameters are adapted. In [23], an ensemble of OS-ELM with a forgetting mechanism (FOS-ELM), which can learn samples one-by-one or in batches/chunks, is presented. In this approach, each data has a period of validity, and old data are continuously discarded from the learning process. As in the OB and EOS-ELM algorithms, only the models are adapted over time. On-line ensembles using a Sliding Window (SW) have been widely adopted for soft computing applications. In [18], a window slides along the training data, and when a change is detected by a *t-test*, a new model is trained using the data window and included into the ensemble. Kaneko and Funatsu [24] design an ensemble of on-line SVM models. During the on-line phase, a new model is added at a fixed frequency using the current data window. In [25] the data is partitioned into different subsets using the Fuzzy C-means cluster algorithm; And each subset is used to train a Least Squares SVM ensemble component model.

However, all these listed methods do not add and remove models over time; but the on-line inclusion and removal of models is an important key for improving ensemble prediction performance. Additive Expert (AddExp) [26], On-line Weighted Ensemble (OWE) [27] and Dynamic and On-line Ensemble Regression (DOER) [28] add new models when the ensemble's error on the newest sample is greater than a threshold, and remove inaccurate models over time; while Online Accuracy Updated Ensemble (OAUE) [29] and Learn^{++}.NSE [21] add and remove models when a batch is available.

Ensemble pruning strategy is a recent topic for changing environments. It refers to the technique employed to select a subset of models from the original set of models, and removing those that are detrimental to the ensemble's prediction performance. In off-line ensembles, a subset of models is usually selected by Genetic Algorithms [30, 31] or Greedy optimizations [32] based on the ensemble error and/or ensemble diversity. However, these methods are computationally expensive for on-line applications, so that other methods should be preferred. The ordered aggregation (OA) technique uses some measure in order to produce a decreasing order of the best models for a given data. In [33] models are ordered according their accuracies on a validation data set; while in [34] models are ranked according to their accuracies and their diversities.

This paper proposes a new on-line sample-based ensemble of ELM models using OA (termed as OEOA) which is able to provide on-line prediction of variables in changing environments. OEOA selects dynamically an optimal subset size of models based on the minimization of the error of the ensemble on the newest sample. Then, the models are ordered based on their on-line prediction errors and the best models of the ordered sequence are employed to obtain the ensemble's output. OEOA builds an ensemble based on a SW. A new model is trained (with samples of the current window) and added, if the current ensemble's error is higher than a threshold. The error of each model is obtained using a window that is filled with its predictive errors on the most recent on-line samples. The models' weights are dynamically assigned according to their prediction errors. Inaccurate models are removed for assuring adaptation of the ensemble in changing environments. As a base model for the ensemble, this paper proposes a new OS-ELM model using DFF (termed as λ_{DFF}OS-ELM) which shows superior accuracy in industrial applications when compared to the well-known OS-ELM model. Experiments are reported to demonstrate the performance and effectiveness of the proposed methods.

The main contributions of this work are: (1) a new on-line sample-based ensemble of regressor models using OA that overcomes the problem of defining the optimal ensemble size; in most cases, the subset model selection obtains better performance than aggregating all the models; (2) a new on-line ELM model using variable FF; results are presented to demonstrate its performance, effectiveness, and faster adaptation capability and accuracy; and (3) thorough analysis of the experimental results using on-line ensembles of the state-of-the-art and OEOA, revealing that their performances can be significantly improved using λ_{DFF}OS-ELM as the base model in industrial data sets.

The paper is organized as follows. Section 2 presents a background on ELM models. Section 3 describes the λ_{DFF}OS-ELM algorithm. Section 4 describes the OEOA algorithm. The experimental results are presented and analyzed in Section 5. Section 6 presents concluding remarks.

2. Background

In this section, background of the ELM and the OS-ELM is introduced. For the sake of simplicity, this paper considers ELM for regression with one single output.

2.1. ELM

Consider a data set $\mathbf{D} = \{(\mathbf{x}_t, y_t) \mid t = 1, \ldots, T\}$ with T distinct samples, where $\mathbf{x}_t \in \mathbb{R}^r$ and $y_t \in \mathbb{R}$. A standard ELM with

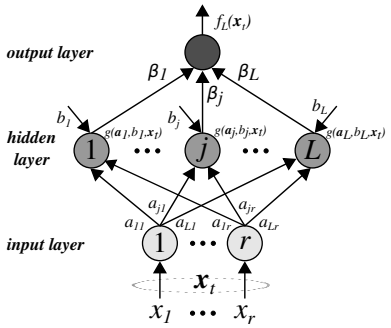


Figure 1: ELM structure.

$L \leq T$ hidden nodes and activation function $g(x)$ (e.g. sigmoid: $g(x) = 1/(1 + \exp(-x))$) will be employed, and is mathematically represented as [35]:

$$f_L(\mathbf{x}_t) = \sum_{j=1}^{L} \beta_j g(\mathbf{a}_j, b_j, \mathbf{x}_t) = o_t, \text{ for } t = 1, \ldots, T, \quad (1)$$

where $\mathbf{a}_j = [a_{j1}, a_{j2}, \ldots, a_{jr}]^{\mathrm{T}}$ is the input weight vector connecting the r input nodes and the j-th hidden node, $j = 1, \ldots, L$; b_j is the bias of the j-th hidden node; β_j connects the j-th hidden node and the output node; and o_t is the predicted output. Figure 1 shows the ELM structure. If an ELM can approximate the T samples of $\mathbf{D}$ with zero error, then Equation (1) can be written as:

$$f_L(\mathbf{x}_t) = \sum_{j=1}^{L} \beta_j g(\mathbf{a}_j, b_j, \mathbf{x}_t) = y_t, \text{ for } t = 1, \ldots, T. \quad (2)$$

The ELM can be represented as:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{y}, \quad (3)$$

$$\mathbf{H} = \begin{bmatrix} g(\mathbf{a}_1, b_1, \mathbf{x}_1) & \ldots & g(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \ldots & \vdots \\ g(\mathbf{a}_1, b_1, \mathbf{x}_T) & \ldots & g(\mathbf{a}_L, b_L, \mathbf{x}_T) \end{bmatrix}_{T \times L}, \quad (4)$$

$$\boldsymbol{\beta} = [\beta_1, \ldots, \beta_L]^{\mathrm{T}}, \text{ and } \mathbf{y} = [y_1, \ldots, y_T]^{\mathrm{T}}, \quad (5)$$

where $\boldsymbol{\beta}$ is the output weight vector and $\mathbf{y}$ is the output vector. $\mathbf{H}$ is called hidden layer output matrix, where the j-th column of $\mathbf{H}$ represents the j-th hidden node output vector with respect to all the inputs; and the t-th row of $\mathbf{H}$ is the output vector of the hidden layer with respect to $\mathbf{x}_t$.

The input weights and biases are randomly assigned, and the learning in ELM is based on finding a solution for the vector $\boldsymbol{\beta}$. In most cases, the number of training samples is greater than the number of hidden neurons (i.e. $T > L$); so that $\mathbf{H}$ is a nonsquare matrix and there may not exist a $\boldsymbol{\beta}$ such that $\mathbf{H}\boldsymbol{\beta} = \mathbf{y}$. Such solution for $\boldsymbol{\beta}$ can be determined using a LS method as:

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{y}, \quad (6)$$

Algorithm 1 Learning Algorithm for ELM Models.

Input: a training data set $\mathbf{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$, including L distinct samples; an activation function $g(x)$; a number of hidden nodes L; (where $L \leq T$);

1. Randomly assign input weights $\mathbf{a}_j$ and biases b_j, $j = 1, \ldots, L$;
2. Obtain matrix $\mathbf{H}$ using $\mathbf{D}$ and Equation (4);
3. Obtain the output weight $\boldsymbol{\beta}$ through Equation (8).

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse or pseudoinverse [36] of matrix $\mathbf{H}$. If the inverse of $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ exists, then $\mathbf{H}^{\dagger}$ can be calculated as

$$\mathbf{H}^{\dagger} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}. \quad (7)$$

Substituting Equation (7) into Equation (6), then $\boldsymbol{\beta}$ becomes:

$$\boldsymbol{\beta} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{y}. \quad (8)$$

From Theorem II.1 of Liang et al. [6], if L training samples in $\mathbf{D}$ are distinct, then rank($\mathbf{H}$) = L, which implies that the inverse of $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ in (8) exists [37]. The ELM algorithm is summarized in Algorithm 1.

2.2. OS-ELM

OS-ELM learning consists of two phases: the *initialization phase* and the *sequential learning phase* [6]. In the initialization phase, an initial training data set, $\mathbf{D}_0 = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T_0}$ from a data set $\mathbf{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$ (with $T_0 < T$), is considered for designing an initial ELM. In the sequential learning phase, on-line samples are employed either one-by-one or on batches/chunks (with fixed or varying size) for on-line retraining of the ELM, where the $(k + 1)$-th chunk of data set is given by:

$$\mathbf{D}_{k+1} = \{(\mathbf{x}_t, y_t)\}_{t=(\sum_{l=0}^{k} T_l)+1}^{t=\sum_{l=0}^{k+1} T_l}, \quad (9)$$

where $k \geq 0$ and T_{k+1} is the number of samples in the $(k + 1)$-th chunk. The initialization phase is similar to the standard ELM learning. The initial output weight vector $\boldsymbol{\beta}_0$ is determined as:

$$\boldsymbol{\beta}_0 = (\mathbf{H}_0^{\mathrm{T}}\mathbf{H}_0)^{-1}\mathbf{H}_0^{\mathrm{T}}\mathbf{y}_0, \quad (10)$$

where $\mathbf{y}_0 = [y_1, \ldots, y_{T_0}]^{\mathrm{T}}$ is the output vector from $\mathbf{D}_0$ (or the initial output vector); and $\mathbf{H}_0$ is the initial hidden layer output matrix obtained with $\mathbf{D}_0$:

$$\mathbf{H}_0 = \begin{bmatrix} g(\mathbf{a}_1, b_1, \mathbf{x}_1) & \ldots & g(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \ldots & \vdots \\ g(\mathbf{a}_1, b_1, \mathbf{x}_{T_0}) & \ldots & g(\mathbf{a}_L, b_L, \mathbf{x}_{T_0}) \end{bmatrix}_{T_0 \times L}. \quad (11)$$

Considering $\mathbf{P}_0 = (\mathbf{H}_0^{\mathrm{T}}\mathbf{H}_0)^{-1}$, where $\mathbf{P}_0$ is the initial covariance matrix, Equation (10) can be written as:

$$\boldsymbol{\beta}_0 = \mathbf{P}_0\mathbf{H}_0^{\mathrm{T}}\mathbf{y}_0. \quad (12)$$

Upon the arrival of $(k + 1)$-th chunk, the new output weight vector $\boldsymbol{\beta}_{k+1}$ is computed using concepts of the RLS algorithm as follows:

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \mathbf{P}_{k+1}\mathbf{H}_{k+1}^{\mathrm{T}}(\mathbf{y}_{k+1} - \mathbf{H}_{k+1}\boldsymbol{\beta}_k), \quad (13)$$

Algorithm 2 Learning Algorithm for the OS-ELM Model.

Input: A data set $\mathbf{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$; an activation function $g(x)$; a number of hidden nodes L; number of samples for the initialization phase T_0, including L distinct samples (where $L \leq T_0 < T$);

1. **Initialization/training phase:** Consider a training data $\mathbf{D}_0 = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T_0}$;
   (a) Randomly assign input weights $\mathbf{a}_j$ and biases b_j, $j = 1, \ldots, L$;
   (b) Calculate $\mathbf{H}_0$ using $\mathbf{D}_0$ and Equation (11);
   (c) Obtain the output weight $\boldsymbol{\beta}_0$ through Equation (12), where $\mathbf{P}_0 = (\mathbf{H}_0^{\mathrm{T}}\mathbf{H}_0)^{-1}$ and $\mathbf{y}_0 = [y_1, \ldots, y_{T_0}]^{\mathrm{T}}$; Set $k = 0$;
2. **Sequential/on-line learning phase:** Present the $(k+1)$-th chunk $\mathbf{D}_{k+1}$ defined in Equation (9);
   (a) Obtain matrix $\mathbf{H}_{k+1}$ using $\mathbf{D}_{k+1}$ and Equation (15);
   (b) Set $\mathbf{y}_{k+1}$ using Equation (16);
   (c) Obtain $\mathbf{P}_{k+1}$ and $\boldsymbol{\beta}_{k+1}$ using Equations (14) and (13), respectively;
   (d) Set $k \leftarrow k+1$; Go to Step 2.

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k\mathbf{H}_{k+1}^{\mathrm{T}}(\mathbf{I} + \mathbf{H}_{k+1}\mathbf{P}_k\mathbf{H}_{k+1}^{\mathrm{T}})^{-1}\mathbf{H}_{k+1}\mathbf{P}_k\,, \tag{14}$$

$$\mathbf{H}_{k+1} = \begin{bmatrix} g(\mathbf{a}_1, b_1, \mathbf{x}_{(\sum_{l=0}^{k} T_l)+1}) \ldots g(\mathbf{a}_L, b_L, \mathbf{x}_{(\sum_{l=0}^{k} T_l)+1}) \\ \vdots \quad \ldots \quad \vdots \\ g(\mathbf{a}_1, b_1, \mathbf{x}_{\sum_{l=0}^{k+1} T_l}) \ldots g(\mathbf{a}_L, b_L, \mathbf{x}_{\sum_{l=0}^{k+1} T_l}) \end{bmatrix}_{T_{k+1}\times L}, \tag{15}$$

$$\mathbf{y}_{k+1} = [y_{(\sum_{l=0}^{k} T_l)+1}), \ldots, y_{\sum_{l=0}^{k+1} T_l}]^{\mathrm{T}}. \tag{16}$$

For detailed derivation of Equations (13) and (14) paper [6] is suggested. In order to make rank$(\mathbf{H}_0) = L$ so that $\mathbf{H}_0^{\mathrm{T}}\mathbf{H}_0$ is invertible, L distinct samples in are assumed to be used in $\mathbf{D}_0$. When the $(k+1)$-th chunk contains only one sample, Equations (13) and (14) can be written using the Sherman-Morrison formula[1] as [38]:

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \mathbf{P}_{k+1}\mathbf{h}_{k+1}(y_{k+1} - \mathbf{h}_{k+1}^{\mathrm{T}}\boldsymbol{\beta}_k), \tag{17}$$

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \frac{\mathbf{P}_k\mathbf{h}_{k+1}\mathbf{h}_{k+1}^{\mathrm{T}}\mathbf{P}_k}{1 + \mathbf{h}_{k+1}^{\mathrm{T}}\mathbf{P}_k\mathbf{h}_{k+1}}, \tag{18}$$

where $\mathbf{h}_{k+1} = [g(\mathbf{a}_1, b_1, \mathbf{x}_{(\sum_{l=0}^{k} T_l)+1}), \ldots, g(\mathbf{a}_L, b_L, \mathbf{x}_{(\sum_{l=0}^{k} T_l)+1})]$. The OS-ELM algorithm is summarized in Algorithm 2.

3. OS-ELM Model with Variable FF (λ_{DFF}OS-ELM)

A new OS-ELM model with variable FF, called λ_{DFF}OS-ELM, is proposed in this section. It is based on the assumption that *a priori* selection of the FF may not be able to track all the system dynamics. The proposed method, λ_{DFF}OS-ELM, incorporates the DFF method, which allows the dynamic and automatic adaptation of the FF depending on the new information of the input and output data. The forgetting is made sensitive to direction of the incoming data. Additionally, the DFF method suppresses obsolete information by modifying only that piece of the so-far accumulated information which is being innovated by the currently incoming data. Moreover, the DFF method avoids the windup phenomenon [13], since the model is adapted only if the new data contains new information; and is useful for systems with time-varying behavior.

Similarly to the OS-ELM algorithm, the λ_{DFF}OS-ELM has two phases: *initialization phase* and *on-line learning phase*. In the initialization phase, a training data set of size T_0, $\mathbf{D}_0 = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T_0}$ from a data set $\mathbf{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$ (with $T_0 < T$), is used to train an initial model. In the on-line learning phase, on-line samples from a data set $\mathbf{D}_{online} = \{(\mathbf{x}_t, y_t)\}_{t=T_0+1}^{T}$ are given incrementally one-by-one for on-line retraining of the model.

In the initialization phase, the initial output weight vector $\boldsymbol{\beta}_0$ is obtained using Equation (12):

$$\boldsymbol{\beta}_0 = \mathbf{P}_0\mathbf{H}_0^{\mathrm{T}}\mathbf{y}_0,$$

where $\mathbf{P}_0 = (\mathbf{H}_0^{\mathrm{T}}\mathbf{H}_0)^{-1}$; $\mathbf{y}_0 = [y_1, \ldots, y_{T_0}]^{\mathrm{T}}$ (from $\mathbf{D}_0$); $\mathbf{H}_0$ is the initial hidden layer output matrix and it is obtained from Equation (11), where L is the number of hidden nodes; and $\boldsymbol{\beta}_0 = [\beta_1, \ldots, \beta_L]^{\mathrm{T}}$.

In the on-line learning phase, when a new sample $(\mathbf{x}_t, y_t)$ from $\mathbf{D}_{online}$ is available, it is employed to obtain a new output weight vector $\boldsymbol{\beta}_{k+1}$ (with $k \geq 0$ and $k = t - T_0 - 1$) using the RLS with DFF [14, 39] as follows:

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \frac{\mathbf{P}_k\mathbf{h}_{k+1}}{1 + \xi_{k+1}}\hat{e}_{k+1}\,, \tag{19}$$

$$\hat{e}_{k+1} = y_t - \mathbf{h}_{k+1}^{\mathrm{T}}\boldsymbol{\beta}_k, \tag{20}$$

$$\xi_{k+1} = \mathbf{h}_{k+1}^{\mathrm{T}}\mathbf{P}_k\mathbf{h}_{k+1}, \tag{21}$$

$$\mathbf{h}_{k+1} = [g(\mathbf{a}_1, b_1, \mathbf{x}_t), \ldots, g(\mathbf{a}_L, b_L, \mathbf{x}_t)], \tag{22}$$

where $\hat{e}_{k+1}$ is the prediction error on the new sample using vector $\boldsymbol{\beta}_k$; ξ_{k+1} is an auxiliary scalar and $\mathbf{h}_{k+1}$ is a hidden layer output vector. If $\xi_{k+1} = 0$, then the new covariance matrix $\mathbf{P}_{k+1}$ is obtained as: $\mathbf{P}_{k+1} = \mathbf{P}_k$. Otherwise, if $\xi_{k+1} > 0$, then $\mathbf{P}_{k+1}$ is obtained as:

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \frac{\mathbf{P}_k\mathbf{h}_{k+1}\mathbf{h}_{k+1}^{\mathrm{T}}\mathbf{P}_k}{\varepsilon_{k+1}^{-1} + \xi_{k+1}}\,, \tag{23}$$

$$\varepsilon_{k+1} = \lambda_k - \frac{1 - \lambda_k}{\xi_{k+1}}, \tag{24}$$

where ε_{k+1} is an auxiliary parameter; and λ_k is the FF at the k-th iteration, and it should be initialized as $0 < \lambda_0 \leq 1$. The smaller the FF, the smaller the influence of the old data to the current model's parameters. The FF for the $(k+1)$-th iteration is obtained as [14, 40, 41]:

$$\lambda_{k+1} = \left\{1 + (1+\rho)\left[\ln(1+\xi_{k+1}) + \left(\frac{(\upsilon_{k+1}+1)\eta_{k+1}}{1+\xi_{k+1}+\eta_{k+1}} - 1\right)\frac{\xi_{k+1}}{1+\xi_{k+1}}\right]\right\}^{-1}, \tag{25}$$

where

$$\eta_{k+1} = \hat{e}_{k+1}^2/\gamma_{k+1}, \tag{26}$$

[1] $(\mathbf{B} + uv^{\mathrm{T}})^{-1} = \mathbf{B}^{-1} - \frac{\mathbf{B}^{-1}uv^{\mathrm{T}}\mathbf{B}^{-1}}{1+v^{\mathrm{T}}\mathbf{B}^{-1}u}$.

Algorithm 3 Learning Algorithm for λ_{DFF}OS-ELM.

Input: A data set $\mathbf{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$; an activation function $g(x)$; a number of hidden nodes L; number of samples for the initialization phase T_0, including L distinct samples (where $L \leq T_0 < T$); λ_0; γ_0; υ_0; ρ;

1. **Initialization/training phase:** Consider a training data set $\mathbf{D}_0 = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T_0}$;
   (a) Randomly assign input weights $\mathbf{a}_j$ and biases b_j, $j = 1, \ldots, L$;
   (b) Calculate matrix $\mathbf{H}_0$ using $\mathbf{D}_0$ and Equation (11);
   (c) Obtain the output weight $\boldsymbol{\beta}_0$ through Equation (12), where $\mathbf{P}_0 = (\mathbf{H}_0^{\mathrm{T}}\mathbf{H}_0)^{-1}$, and $\mathbf{y}_0 = [y_1, \ldots, y_{T_0}]^{\mathrm{T}}$;
2. **On-line learning phase:** Consider an on-line data set $\mathbf{D}_{online} = \{(\mathbf{x}_t, y_t)\}_{t=T_0+1}^{T}$; set $t = T_0$;
   (a) **While** $t < T$ **do:**
      i. Set $t \leftarrow t + 1$; $k = t - T_0 - 1$;
      ii. Obtain sample $(\mathbf{x}_t, y_t)$ from $\mathbf{D}_{online}$;
      iii. Obtain vector $\mathbf{h}_{k+1}$ using Equation (22);
      iv. Obtain $\boldsymbol{\beta}_{k+1}$, $\hat{e}_{k+1}$, and ξ_{k+1} using Equations (19)-(21), respectively;
      v. Obtain $\mathbf{P}_{k+1}$:

$$\mathbf{P}_{k+1} = \begin{cases} \mathbf{P}_k, & \text{if } \xi_{k+1} = 0, \\ \text{as Equations (23)-(24)}, & \text{if } \xi_{k+1} > 0; \end{cases}$$

      vi. Calculate η_{k+1}, γ_{k+1}, and υ_{k+1} using Equations (26)-(28), respectively;
      vii. Compute λ_{k+1} using Equation (25);
   (b) **end while**

$$\gamma_{k+1} = \lambda_k \left(\gamma_k + \frac{\hat{e}_{k+1}^2}{1 + \xi_{k+1}} \right), \tag{27}$$

$$\upsilon_{k+1} = \lambda_k (\upsilon_k + 1), \tag{28}$$

ρ is a positive constant; and η_{k+1}, γ_{k+1}, and υ_{k+1} are auxiliary parameters. The initial values of γ and υ (i.e. γ_0 and υ_0) should be set between 0 and 1. The λ_{DFF}OS-ELM learning algorithm is summarized in Algorithm 3. Similarly to the OS-ELM algorithm, it is assumed that L distinct samples, with $L \leq T_0 < T$, are included among the m samples contained in $\mathbf{D}_0$.

It should be pointed out that, since the approaches available in [14, 40, 41] perform parameter updates on a sample basis, then the λ_{DFF}OS-ELM algorithm can only be adapted at the sample basis. Additional work is envisaged to develop a new λ_{DFF}OS-ELM algorithm that can also be adapted at the batch basis.

4. An On-line Ensemble Using Ordered Aggregation (OEOA)

OEOA designs an ensemble using a SW. It employs the common assumption that the most recent data provides the best and most relevant representation of the current state of the process and of the near-future state; thus only this data should be kept [29, 42]. A data window of fixed size is kept, and when a new sample is available, it is added to the window, and the oldest sample is removed from the window. The data window is employed to train a new model when the ensemble's performance is deteriorating, and to obtain the models' prediction errors. The main strategies of OEOA for achieving faster adaptivity in time-varying environments are: *sample-based ensemble* which offers higher performance and faster adaptivity when compared to batch-based ensembles; *models' weights adaptation*; *models' parameters adaptation*; and *dynamic inclusion and removal of models*.

Firstly, consider a regression problem with a data set $\mathbf{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$, where $\mathbf{x}_t \in \mathbb{R}^r$ and $y_t \in \mathbb{R}$, and a window's size T_0 (with $T_0 < T$). Consider an ensemble $\mathcal{E}$ with M_{max} models, where $\mathcal{E} = \{f_1, \ldots, f_{M_{max}}\}$ and $f_m \in \mathcal{E}$ represents a model. OEOA has two phases: creation of an initial pool of M_{max} models, and on-line learning phase. In the first phase, an initial data window $\mathbf{D}^t$ with the first T_0 samples of $\mathbf{D}$ is employed to train the initial pool of models. In the second phase, samples $t = T_0 + 1, \ldots, T$ from $\mathbf{D}$ are given one-by-one for on-line prediction and on-line learning. Therefore, for each t, a data window $\mathbf{D}^t$ keeps the most recent T_0 samples.

Before introducing the OEOA algorithm, Section 4.1 describes the models' characteristics. Then, Section 4.2 details the OEOA algorithm.

4.1. OEOA Component Models

Each model f_m from ensemble $\mathcal{E}$ is initially trained with samples from a data window $\mathbf{D}^t$ using the initialization phase of an on-line supervised learner (e.g. OS-ELM or λ_{DFF}OS-ELM). The main parameters associated to f_m are: $life_m$ which denotes the total number of on-line predictions performed by f_m; w_m which is the weight of model f_m; and MSE_m^t which denotes the total prediction error of f_m at time t.

When a model f_m predicts a sample $(\mathbf{x}_t, y_t)$, its prediction error e_m^t is determined as $e_m^t = (y_t - \hat{y}_t^m)^2$, where $\hat{y}_t^m$ is the output predicted by model f_m. At each time t, MSE_m^t is obtained as:

$$\mathrm{MSE}_m^t = \begin{cases} 0, & \text{if } life_m = 0, \\ \frac{life_m - 1}{life_m} \cdot \mathrm{MSE}_m^{t-1} + \frac{1}{life_m} \cdot e_m^t, & \text{if } 1 \leq life_m \leq T_0, \\ \mathrm{MSE}_m^{t-1} + \frac{e_m^t}{T_0} - \frac{e_m^{t-T_0}}{T_0}, & \text{if } life_m > T_0. \end{cases} \tag{29}$$

This approach is similar to one proposed by OAUE [28, 29]. The aim is to estimate the average of the predictive error of f_m on the most recent T_0 samples using the Mean Squared Error (MSE). Equation (29) works like an adaptive MSE. A new model initially receives MSE_m^t equal to 0. As it performs on-line predictions and the variable life $life_m$ is incremented, the window of errors is also enlarged up to a maximum width T_0. If $life_m > T_0$ at a time t, the new error e_m^t is considered to compute MSE_m^t and the old error $e_m^{t-T_0}$ is eliminated in the calculation of MSE_m^t. Note that only errors observed on the on-line phase are considered to calculate MSE_m^t.

4.2. OEOA Algorithm Description

OEOA selects a subset of its models to participate in forming the ensemble prediction. Ensemble model selection usually

involves selecting an optimal subset of models by searching the space of all models' combinations. However, the computational complexity of such an approach is exponential in the number of models: an ensemble with M_{max} models involves searching a space of $(2^{M_{max}} - 1)$ non-empty solutions to minimize a cost function. OEOA sorts models according to their errors obtained on the on-line predictions to avoid exhaustive search. Then, the best B models (with $B \leq M_{max}$) in the ordered sequence are selected as the optimal subset of models for predicting each incoming sample. When the real output is available, the optimal subset size is determined so as to minimize the ensemble prediction error on the newest sample.

The proposed OEOA method is summarized in Algorithm 4. Factor α controls the inclusion of a new model based on the prediction error on the newest sample [28, 27], where $0 < \alpha < 1$. An ensemble $\mathcal{E} = \{f_1, \dots f_{M_{max}}\}$ with $M_{max} > 1$ models is considered. To avoid the problem of reaching a small size of the subset of models [43], a variable M_{min} was included to control the minimum size of the optimal subset in the ordered aggregation, where $1 < M_{min} \leq M_{max}$.

In Step 2 a pool of M_{max} models is created using the initialization phase of a generic on-line supervised learner. The models are trained using the initial window $\mathbf{D}^t = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T_0} \subset \mathbf{D}$. When a new sample $(\mathbf{x}_t, y_t)$ is available (Step 5a), the window slides along the data (Step 5b). This operation adds the new sample $(\mathbf{x}_t, y_t)$ to the window and excludes the oldest sample $(\mathbf{x}_{t-T_0}, y_{t-T_0})$ from the window. In Step 5d, the final output of the optimal subset of B models is given. It is obtained by a weighted sum of the models' outputs. This step is performed using the Algorithm 5, an algorithm that obtains the output prediction based on the ordered aggregation of the best models.

Algorithm 5 obtains a vector of indexes, $\mathbf{IX}_{Top} = [ix_1, \dots ix_Q]$, of the Q best performing models of the ensemble with respect to the $\mathbf{MSE}^{t-1}$. The MSE values of this subset of models are kept in vector $\mathbf{MSE}_{Top}^{t-1}$. Step 3 of Algorithm 5 aims to obtain only the weights of the subset of models. Equation (30) transforms weights in such a way that a model f_m with MSE_m^{t-1} around the median value receives a weight equal to 1. Models with MSE_m^{t-1} lower than the median have their weights exponentially increased, while models with MSE_m^{t-1} larger than the median have their weights exponentially decreased. Therefore, more "credit" is given to the models that have high accuracy. A model with $life_m = 0$ created at time t is initialized with weight equal to 1. This criterion smooths the contribution of a new model at the time $t + 1$, the time at which such model will be evaluated on-line for the first time.

Then, the question is how to determine the optimal subset size B for the next iteration. B is chosen so as to minimize the square error on the newest sample $(\mathbf{x}_t, y_t)$:

$$B = \underset{p=M_{min},\dots,M_{max}}{\text{argmin}} \left(\epsilon^p\right), \tag{31}$$

where $\epsilon^p = (y_t - \hat{o}_p)^2$, and $\hat{o}_p$ is the output prediction of an ordered aggregation with the best p models of the ensemble with respect to the $\mathbf{MSE}^{t-1}$. This strategy may obtain a small value of B and induce the inclusion of only new models, since a new model f_m is initialized with $\text{MSE}_m^t = 0$. To prevent this case and assure stability to the ensemble, M_{min} should be large

Algorithm 4 Learning Algorithm for OEOA.

Input: A data set $\mathbf{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$; window's size, T_0; an on-line supervised learner; α, factor to add a new model; maximum and minimum number of models, M_{max} and M_{min} respectively, (where $1 < M_{min} \leq M_{max}$);

Creating a pool of M_{max} models:

1. Set $\mathcal{E} \leftarrow \emptyset$; $t = T_0$; number of considered best models $B = M_{max}$; current window $\mathbf{D}^t = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T_0} \subset \mathbf{D}$;
2. **for** $m = 1, \dots M_{max}$ **do**:
   (a) $f_m \leftarrow$ Obtain a new model trained with $\mathbf{D}^t$ using the initialization phase of the on-line supervised learner (e.g. λ_{DFF}OS-ELM);
   (b) Set $life_m = 0$, $\text{MSE}_m^t = 0$, and $w_m = 1$;
   (c) Include f_m into the ensemble: $\mathcal{E} \leftarrow \mathcal{E} + \{f_m\}$;
3. **end for**
4. Build the vector of the MSE of the models: $\mathbf{MSE}^t = [\text{MSE}_1^t, \dots, \text{MSE}_{M_{max}}^t]$;

On-line learning phase:

5. **for** $t = T_0 + 1, \dots, T$ **do**:
   (a) Receive a new sample $(\mathbf{x}_t, y_t)$;
   (b) Slide the window: $\mathbf{D}^t = \mathbf{D}^{t-1} + (\mathbf{x}_t, y_t) - (\mathbf{x}_{t-T_0}, y_{t-T_0})$;
   (c) Get models' predictions $\hat{\mathbf{y}}_t$, where $\hat{\mathbf{y}}_t = [\hat{y}_t^1, \dots, \hat{y}_t^{M_{max}}]$ and $\hat{y}_t^m = f_m(\mathbf{x}_t)$, for $m = 1, \dots, M_{max}$;
   (d) Obtain the final prediction of the optimal subset: $F(\mathbf{x}_t) \leftarrow \text{OutputOEOA}(\mathcal{E}, M_{max}, B, \mathbf{MSE}^{t-1}, \hat{\mathbf{y}}_t)$;
   (e) **If** $M_{min} \neq M_{max}$, **then** Determine a new value for B:
   Set minError = ∞;
   for $p = M_{min}, \dots, M_{max}$ **do:**
      i. $\hat{o}_p \leftarrow \text{OutputOEOA}(\mathcal{E}, M_{max}, p, \mathbf{MSE}^{t-1}, \hat{\mathbf{y}}_t)$;
      ii. Determine the error as $\epsilon^p = (y_t - \hat{o}_p)^2$;
      iii. **if** $\epsilon^p <$ minError, **then** Set minError = ϵ^p; $B = p$;
   end for
   (f) Update the models (for $m = 1, \dots, M_{max}$):
      i. Obtain the error e_m^t of f_m for input $\mathbf{x}_t$: $e_m^t = (y_t - \hat{y}_t^m)^2$;
      ii. Set $life_m \leftarrow life_m + 1$;
      iii. Obtain MSE_m^t using Equation (29);
      iv. Incrementally retrain model f_m using sample $(\mathbf{x}_t, y_t)$ and using (one iteration of) the on-line learning phase of the on-line supervised learner (e.g. λ_{DFF}OS-ELM);
   (g) Build vector $\mathbf{MSE}^t = [\text{MSE}_1^t, \dots, \text{MSE}_{M_{max}}^t]$;
   (h) **if** $|(F(\mathbf{x}_t) - y_t)/y_t| > \alpha$
      i. $f_0 \leftarrow$ Obtain a new model trained with $\mathbf{D}^t$ using the initialization phase of the on-line supervised learner;
      ii. Set $life_0 = 0$, $\text{MSE}_0^t = 0$, and $w_0 = 1$;
      iii. Replace model $f_z \in \mathcal{E}$ by f_0: where $z = \text{argmax}_{n=1,\dots,N_{max}}(\text{MSE}_n^t)$, $f_z \leftarrow f_0$ and $life_z \leftarrow life_0$;
6. **end for**

Algorithm 5 OutputOEOA: output prediction based on the ordered aggregation of the best models.

Input: Ensemble $\mathcal{E}$; M_{max}; Q; $\mathbf{MSE}^{t-1}$; $\hat{\mathbf{y}}_t$;

1. Sort the elements of $\mathbf{MSE}^{t-1}$ in ascending order forming $\mathbf{MSE}^{t-1}_{Sort} = [\mathrm{MSE}^{t-1}_{ix_1}, \dots \mathrm{MSE}^{t-1}_{ix_{M_{max}}}]$ and return a vector of indexes $\mathbf{IX}_{Sort} = [ix_1, \dots ix_{M_{max}}]$ which contains the position of each element of $\mathbf{MSE}^{t-1}_{Sort}$ in vector $\mathbf{MSE}^{t-1}$;
2. Assign to $\mathbf{MSE}^{t-1}_{Top} = [\mathrm{MSE}^{t-1}_{ix_1}, \dots \mathrm{MSE}^{t-1}_{ix_Q}]$ and $\mathbf{IX}_{Top} = [ix_1, \dots ix_Q]$ the first Q elements from $\mathbf{MSE}^{t-1}_{Sort}$ and $\mathbf{IX}_{Sort}$, respectively;
3. **for** each $m \in \mathbf{IX}_{Top}$ **do:**

$$w_m = \begin{cases} 1, & \text{if } life_m = 0, \\ \exp\left(-\frac{\mathrm{MSE}^{t-1}_m - \mathrm{median}\left(\mathbf{MSE}^{t-1}_{Top}\right)}{\mathrm{median}\left(\mathbf{MSE}^{t-1}_{Top}\right)}\right), & \text{if } life_m > 0; \end{cases} \quad (30)$$

4. **end for**
5. Obtain the output prediction: $\hat{o}_{Top} = \left(\sum_{m \in \mathbf{IX}_{Top}} w_m \hat{y}^m_t\right) / \sum_{m \in \mathbf{IX}_{Top}} w_m$;

Output: $\hat{o}_{Top}$;

(> 5). It is worth noting that when $M_{min} = M_{max}$ no ordered strategy is employed, and the ensemble has a fixed number of models, since $B = M_{max}$ in all iterations. This strategy is called as "OEOA without ordering". In this case, the algorithm has lower processing time. However, the main advantage of the "OEOA with ordering" (specifically the main advantage of having $M_{min} \neq M_{max}$) is that it is not necessary to tune the ensemble size (but only M_{min} and M_{max}), and then the algorithm dynamically selects the optimal ensemble size; and in some cases, this strategy has higher accuracy when compared to ensembles of fixed size.

In Step 5f the parameters of all models are updated. All the models are retrained, keeping the models updated on the current state of the process. Step 5h evaluates if a new model should be added. The criterion adds a new model when the absolute relative error of the ensemble on the newest sample is greater than α. The new model f_0 is trained using the samples from the current data window $\mathbf{D}^t$; and f_0 replaces the least accurate model of the ensemble. The criterion substitutes the model f_z with the highest error, MSE^t_z. A new model created at iteration t is never excluded by the pruning strategy at the same time t.

5. Experimental Results

In this section, four artificial data sets and five real-world data sets with time-varying behavior are employed to demonstrate the predictive performance of λ_{DFF}OS-ELM and OEOA over state-of-the-art approaches. Artificial data sets allow the control of relevant data set parameters and to perform empirical evaluation of the algorithms in several types of changes. In the tests, it is used the hyperplane data set proposed in [26], a benchmark for evaluating algorithms that deal with concept drifts; and the drifting Friedman's function proposed by Ikonomovska [44], a recent benchmark created for evaluating regression algorithms in changing environments. The real-world data sets enable the evaluation of the merit of the proposed approaches in real-world problems, and comparing them with the most recent works in real-world problems. This paper employs well-known industrial data sets widely used to evaluate algorithms for dynamic system modeling. The experiments have been performed on the Matlab environment, running on a PC equipped with an Intel Core i7-4700MQ 2.4GHz-3.4GHz processor of 4 cores and 8GB of RAM.

5.1. Data Set Description

Artificial data sets. The hyperplane data set involves noise, gradual drift and non-recurring drift [26]. It has 10 input variables, 1 output variable and 2000 samples. The data set has 4 concepts, and 3 concept changes, where each concept holds 500 samples. The Friedman's function is a benchmark for generating data [45]. It employs linear and non-linear relations between input and output variables. The function has a total of 10 input variables and 1 output. To simulate time-varying scenarios, 3 data sets with 2000 samples using the Friedman's function were produced. The first data set, the *local and abrupt drift data set* (Friedman-LA), contains changes in two different regions of the input space using 3 points of abrupt changes. The second data set, the *global recurring abrupt drift data set* (Friedman-GRA), simulates global, abrupt, and recurring drifts using 2 drift points. The third data set, the *global non-recurring gradual drift data set* (Friedman-GnRG), contains 2 episodes of gradual changes. The complete description of these data sets can be found in [44].

Industrial data sets. Six real-world data sets are considered in the experiments, as detailed in Table 1. Most industrial processes exhibit some kind of time-varying behavior, and so these data sets are crucial to evaluate the proposed methodologies. The cement kiln data set was obtained in a real-world environment of a cement plant, where the samples of input variable (e.g. temperatures, pressures, concentrations, etc.) were recorded with a sampling interval of $T = 1$ [min], while the output samples were obtained with different sampling intervals using a laboratory automation system. Information about the other data sets can be found on their corresponding references [18, 46, 47]. The Sulfur Recovery Unit (SRU) data set [47] is a large data set with two outputs. In this paper, only one output is used: the H_2O concentration. Selection of input variables was performed to remove variables with missing values and noises; and to select input variables highly correlated to the output [47].

5.2. Evaluation Methodology

The following methodology is performed to evaluate all the approaches. Consider a data set $\mathbf{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$ with T samples. The single model, the first model of the ensemble, or the pool of models (depending on each approach) is designed using the first T_0 samples from $\mathbf{D}$ in the initialization phase of the learner. The remaining $(T - T_0)$ samples of $\mathbf{D}$ are arranged to form the on-line data to simulate an on-line scenario, where samples are given incrementally one-by-one. Each approach is

Table 1: Specifications of the real-world data sets used in the experiments.

Data set	Output Description	# Samples	# Inputs (bef. pre-proc.)	# Inputs (af. pre-proc.)	Data Set Size
Polymerization reactor[1] [18]	catalyst activity	**648**	15	**10**	*small*
Cement kiln process[2]	free lime (CaO)	**701**	195	**45**	*small*
Powder detergent production[1] [46]	powder specific weight	**1962**	14	**14**	*medium*
Thermal oxidizer[1] [46]	NO_x concentration	**2053**	39	**39**	*medium*
Debutanizer column[3] [47]	butane concentration	**2394**	7	**7**	*medium*
Sulfur recovery unit (SRU)[3] [47]	H_2O concentration	**10081**	5	**5**	*large*

[1] The data set can be made available for academic purposes by requesting it to the authors.
[2] Provided by "AControl - Automação e Controle Industrial, Lda.", Coimbra, Portugal.
[3] `http://www.springer.com/engineering/control/book/978-1-84628-479-3`

evaluated using the mean and standard deviation of the MSE the predicted outputs and the real outputs on the on-line data in 20 trials. In the experiments below, only the MSE on the on-line data is reported.

5.3. Approach Description and Setup

Tests are performed by comparing λ_{DFF}OS-ELM and OEOA to other state-of-the-art methods. The accuracies of the following single model learning algorithms are compared:

- *ELM*: standard ELM [35], implemented as Algorithm 1.
- *OS-ELM*: sample-based OS-ELM [6], implemented as Algorithm 2.
- λ_{DFF}*OS-ELM*: OS-ELM using a variable FF, implemented as Algorithm 3, where $\lambda_0 = 1$; $\gamma_0 = 10^{-3}$; $\upsilon_0 = 10^{-6}$, and $\rho = 0.99$ (the parameters are set as recommend by Bobál et al. [14]).

For each single-model and for each component-model of an ensemble, the hidden layer activation function $g(x)$ is *sigmoid*; and the number of neurons in the hidden layer L is selected by varying it in the interval of $[1, 20]$. This interval may be adjusted to $[1, T_0]$ if $T_0 < 20$, since L should not be greater than T_0 in order to comply with the assumptions in the ELM algorithms. The value of L is selected based on the best performance on a *10-fold cross-validation* using the training data set in 1 trial, where the best number of neurons is selected as the one that maximizes the mean testing performance on the 10-folds. Each OS-ELM or λ_{DFF}OS-ELM model is created by firstly training it with T_0 samples (e.g. belonging to $\mathbf{D}_0$, for OS-ELM or λ_{DFF}OS-ELM, or belonging to $\mathbf{D}_t$ in OEOA) using the initialization phase of the learner; and then, whenever a new on-line sample is available, the model is retrained using the on-line learning phase of the learner.

Experiments are also conduced by comparing the effectiveness of the following on-line ensemble learning algorithms:

- *AddExp*. The AddExp's parameters are: β, a decreasing factor for the models' weights; τ, a factor to include a new model; and γ, a factor to set a new model's weight. They are set based on studies from [26]: $\beta = 0.5$, $\gamma = 0.1$, and $\tau = 0.05$. AddExp was implemented according to [27].
- *DOER*. It was implemented according to [28]. The window's size is T_0. The factor to include a new model (α) will be discussed in Section 5.5.
- *EOS-ELM*. The training data set $\mathbf{D}_0$ has T_0 samples (window). All the models are trained with the same $g(x)$ and L [17], where L is selected as the most frequent best number of neurons on 20 trials of *10-fold cross-validation* on $\mathbf{D}_0$, and at each trial the best number of neurons is selected as the one that maximizes the mean testing performance on the 10-folds. On the on-line phase, all the models are retrained and combined by average.
- *FOS-ELM*. The training data set $\mathbf{D}_0$ has T_0 samples (chunk/block), and $g(x)$ and L for all the models are selected as in the EOS-ELM approach. On the on-line phase, when a chunk of size T_0 is available, the models are updated and combined by average. FOS-ELM was implemented as a batch-based ensemble, as in [23]. The timeless parameter s (a parameter related to the number of last chunks employed to compute the OS-ELM models' parameters) is set according the best results presented in [23]: $s = 4$.
- *Learn*$^{++}$*.NSE*. It was implemented and adapted to regression using the AdaBoost.RT [48], as proposed in [27], where the factor to demarcate incorrect and correct predictions are set to 0.05. Each batch is considered to have size T_0. The slope parameters of the weighing function are set according to the Learn^{++}.NSE authors' suggestions [20]: $a = 0.5$ and $b = 10$.
- *OAUE*. It was implemented according to [29], where each batch/block is considered to have size T_0.
- *OB*. It was implemented according to [22], where L can be different for each model. The training data set has T_0 samples. On the on-line phase, all models are retrained and combined by average.
- *OEOA*. It was implemented according to Algorithm 4. The parameters setting will be discussed in Section 5.5.
- *OWE*. It was implemented according to [27], where the factor to demarcate incorrect and correct predictions is set to $\theta = 0.05$, and the pruning activation factor ρ is set to 1. The window's size is T_0. The discount factor κ is set to 0.2 for all data sets, except for Friedman-GRA where κ is set to 0.99, because it has a recurring nature. The factor to include a new model (α) will be discussed in Section 5.5.

As Learn^{++}.NSE and OWE do not employ model retraining, their base model is the ELM. For the other ensembles (except FOS-ELM), tests are performed using λ_{DFF}OS-ELM and OS-ELM as base models. FOS-ELM has itself a base model: an OS-ELM with a forgetting mechanism which discards old data. For all learning algorithms, on-line data scaling to zero-mean

and unit-variance is performed on the input and output data, where on the on-line scenario the mean and standard deviation of each variable is recursively adapted as new samples are available [49]. As the AddExp requires the presented output data to be normalized to the interval of [0, 1], the outputs of all the data sets are normalized to this interval. In all experiments, this paper considers small values of T_0 (training data set size), since in real-world setups of Soft Sensor applications, it is often difficult to get sufficient data for modeling.

5.4. Comparison of Single Model Learning Algorithms

λ_{DFF}OS-ELM is evaluated and compared to ELM and OS-ELM. For each model, the results are averaged over 20 trials. It has been observed that, for medium size datasets, small windows (e.g. $T_0 = 10$) lead to a significant increase in computational time, and in some cases no improvement in the accuracy of the system is observed. Thus, large windows were chosen for artificial data sets and real-world data sets of medium size. On the other hand, previous tests in [28] have shown that, in real-world data sets of small size, better performance is obtained when small windows are selected; and in real-world data sets of large size, low computational time is obtained when large windows are selected. Therefore, the experiments are conducted by varying T_0 from 20 to 100 in steps of 1 for artificial data sets and real-world data sets of medium size; varying T_0 from 10 to 50 in steps of 1 for real-world data sets of small size; and varying T_0 from 30 to 150 in steps of 1 for the real-world data set of large size (see Table 1). Figure 2 shows the MSE results of each algorithm as a function of T_0 in all data sets. Table 2 shows the average and standard deviation of the MSE and processing time over all values of T_0 for all the algorithms. The processing time considers the time spent on the training and on-line phases.

As observed in Figure 2, for the artificial data sets, the algorithms tend to decrease their errors as T_0 increases. For the real-world data sets, in most cases, OS-ELM and λ_{DFF}OS-ELM methods keep their performances as T_0 increases. For the Friedman data sets, it is observed that OS-ELM outperforms λ_{DFF}OS-ELM. This reveals that λ_{DFF}OS-ELM may not track scenarios with local and abrupt drifts (Friedman-LA), global recurring abrupt drifts (Friedman-GRA) and global non-recurring gradual drifts (Friedman-GnRG). This is because, λ_{DFF}OS-ELM forgets old information over time. λ_{DFF}OS-ELM has the best performance in the hyperplane data set that contains non-recurring abrupt drift. For the real-world data sets, λ_{DFF}OS-ELM obtained the lowest MSE. From Table 2, it is noted that, in terms of processing time, λ_{DFF}OS-ELM outperforms OS-ELM and ELM in most cases.

Table 2: Average and standard deviation of the MSE[1] and processing time (seconds) of the single model learning algorithms by varying T_0.

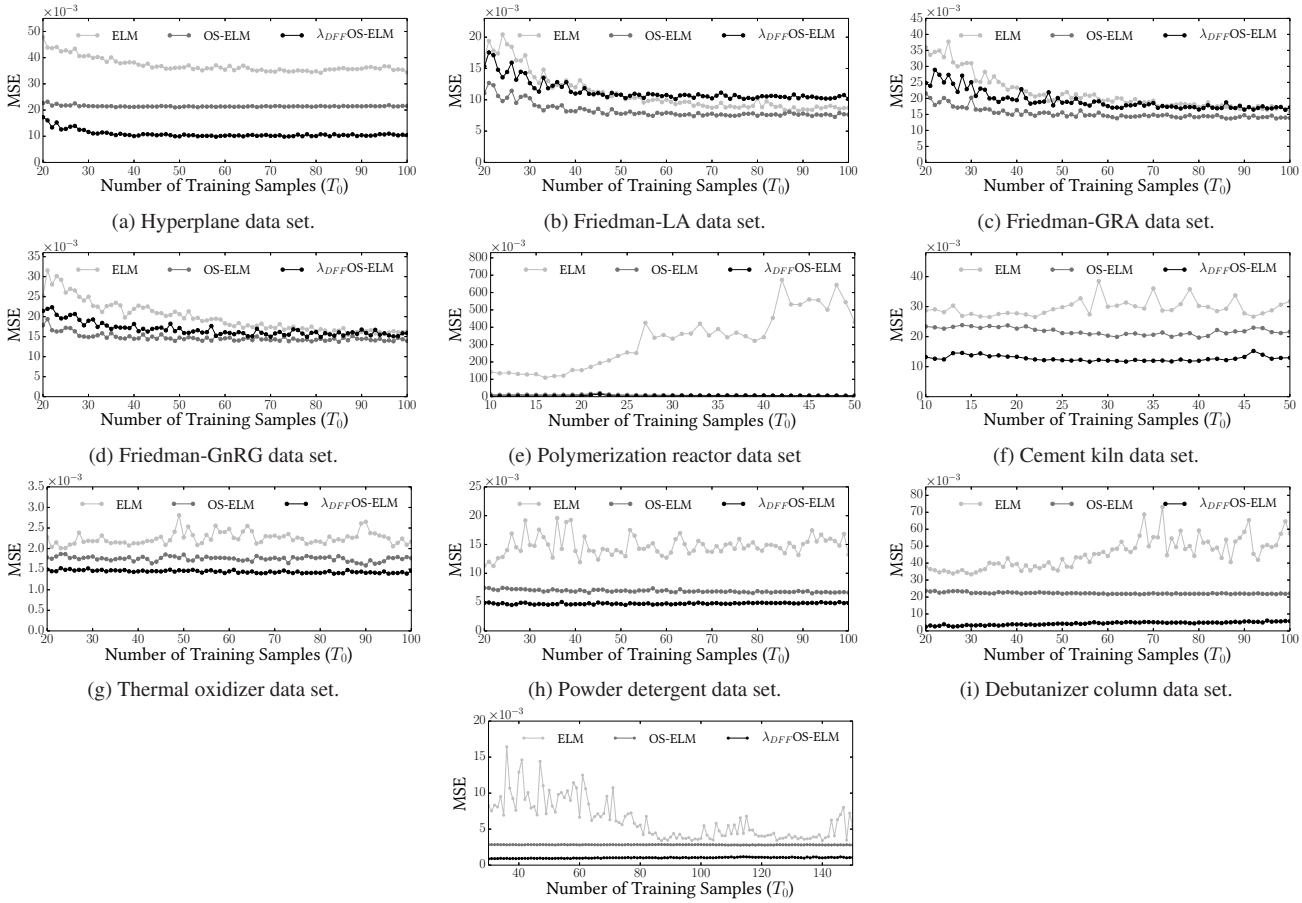
Data set	Approach	MSE	Proc. Time (s)
Hyperplane[2]	ELM	37.120 (2.741)	1.481 (0.259)
	OS-ELM	21.472 (0.323)	1.950 (0.241)
	λ_{DFF}OS-ELM	**10.867 (1.350)**	**1.408 (0.202)**
Friedman-LA[2]	ELM	11.032 (3.024)	1.477 (0.015)
	OS-ELM	**8.285 (1.151)**	2.007 (0.197)
	λ_{DFF}OS-ELM	11.359 (1.618)	**1.332 (0.161)**
Friedman-GRA[2]	ELM	21.497 (5.349)	1.487 (0.016)
	OS-ELM	**15.299 (1.654)**	1.518 (0.208)
	λ_{DFF}OS-ELM	19.253 (2.921)	**1.294 (0.197)**
Friedman-GnRG[2]	ELM	19.735 (3.738)	1.475 (0.013)
	OS-ELM	**14.775 (0.965)**	2.035 (0.176)
	λ_{DFF}OS-ELM	16.801 (1.683)	**1.411 (0.159)**
Polymerization reactor[3]	ELM	323.506 (160.943)	**0.710 (0.075)**
	OS-ELM	9.152 (2.983)	0.878 (0.100)
	λ_{DFF}OS-ELM	**4.891 (2.015)**	0.715 (0.112)
Cement kiln[3]	ELM	29.608 (2.666)	0.757 (0.092)
	OS-ELM	21.825 (1.192)	0.888 (0.106)
	λ_{DFF}OS-ELM	**12.736 (0.882)**	**0.666 (0.099)**
Powder detergent[2]	ELM	14.751 (1.699)	1.473 (0.035)
	OS-ELM	6.922 (0.218)	1.660 (0.314)
	λ_{DFF}OS-ELM	**4.756 (0.119)**	**1.457 (0.145)**
Thermal oxidizer[2]	ELM	2.250 (0.148)	1.788 (0.326)
	OS-ELM	1.755 (0.055)	2.071 (0.298)
	λ_{DFF}OS-ELM	**1.443 (0.031)**	**1.518 (0.154)**
Debutanizer column[2]	ELM	45.426 (8.936)	**1.629 (0.337)**
	OS-ELM	22.243 (0.493)	1.688 (0.322)
	λ_{DFF}OS-ELM	**4.398 (0.883)**	1.693 (0.175)
SRU[4]	ELM	6.236 (2.797)	**5.988 (1.573)**
	OS-ELM	2.827 (0.021)	6.777 (1.504)
	λ_{DFF}OS-ELM	**1.020 (0.066)**	7.455 (2.350)

[1]The values have been multiplied by 10^3.
[2]The values of T_0 are varied from 20 to 100 (in steps of 1).
[3]The values of T_0 are varied from 10 to 50 (in steps of 1).
[4]The values of T_0 are varied from 30 to 150 (in steps of 1).

5.5. Analysis of OEOA parameters

The frequency of adding new models (which is related to α) may impact on the performances of OEOA, OWE, and DOER: small values of α generate large numbers of new models and increase the computational time; and large values of α may produce an inaccurate ensemble in changing environments, since new models are rarely added to the ensemble. Previous tests in [28] were conducted by varying α from 0.04 to 0.1 in steps of 0.02. It has been shown that α is related to the rate of concept change. That is, in data sets where concepts have large sizes (e.g. hyperplane data, where each concept has 500 samples), α should be set to a large value. While in data sets with concepts of small sizes (e.g. most industrial data sets due to the dynamics), α should be set to a small value. Therefore, for the OEOA, OWE, and DOER, in this paper, α is set to 0.10 for the artificial data sets, and 0.04 for the real-world data sets. The training size T_0 (or window size) is also related to the rate of change of the data. In data sets which have a large rate of change, the system has better accuracy when T_0 is small; while in data sets which have a small rate of change, the system has better accuracy when T_0 is large. This characteristic can be observed in the experiments of the next Section.

Experiments were done to evaluate the effect of the minimum number of models (M_{min}) and maximum number of models (M_{max}) in the OEOA algorithm. The experiments use the cement kiln data set with $T_0 = 10$ and $\alpha = 0.04$. The base models are λ_{DFF}OS-ELM and OS-ELM. The ELM model is not used, since it does not have retraining. The first test aims to show the OEOA algorithm when $M_{min} = M_{max}$, namely, no OA is employed (see Figure (3a)). The test reveals that when λ_{DFF}OS-ELM is the base model, the error is reduced as the



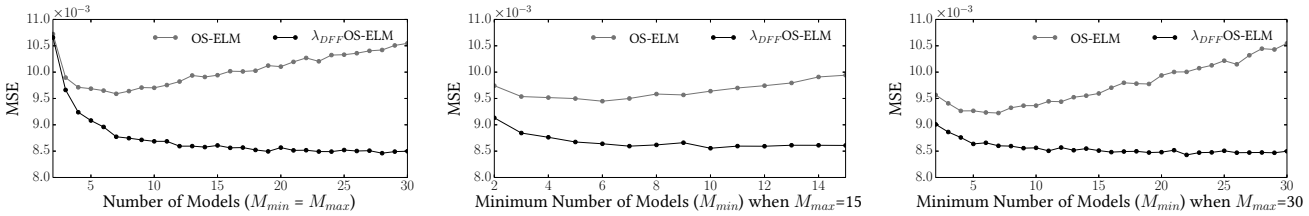
(a) Hyperplane data set. (b) Friedman-LA data set. (c) Friedman-GRA data set.

(d) Friedman-GnRG data set. (e) Polymerization reactor data set (f) Cement kiln data set.

(g) Thermal oxidizer data set. (h) Powder detergent data set. (i) Debutanizer column data set.

(j) SRU data set.

Figure 2: Performance of the single learning algorithms when the number of training samples T_0 increases.



(a) (b) (c)

Figure 3: Experiments using different values of M_{min} and M_{max} in the OEOA algorithm for the cement kiln data set.

number of models increases. When OS-ELM is the base model, the best performances are not obtained with the largest ensemble sizes. Figures (3b) and (3c) show the OEOA's performance when M_{min} varies and M_{max} is fixed. In Figure (3b), M_{max} is set to 15, and in Figure (3c), M_{max} is set to 30. When λ_{DFF}OS-ELM is the base model, as an overall tendency the experiment shows that if M_{min} increases, then the ensemble accuracy increases; while when OS-ELM is the base model, if M_{min} increases the ensemble error increases, after having obtained the best accuracy for some value of M_{min}. Thus, the adequate setting of M_{min} may depend on the base model. The test shows that λ_{DFF}OS-ELM outperforms OS-ELM as base model for the cement kiln data set. This is because, λ_{DFF}OS-ELM models are able to forget old information and track better the dynamics of this data set.

5.6. Comparison of On-line Ensemble Learning Algorithms

In this section, results of the on-line ensemble learning algorithms are compared. For the ensembles of fixed size (i.e. all ensembles except OEOA with the ordering strategy), the maximum number of models (M_{max}) is set to 15. This choice was considered the best suitable for all the ensembles, since in literature M_{max} usually varies between 15 and 30 [20, 26, 28], and the processing time of the experiments increases as M_{max} increases. The OEOA is tested in two scenarios. In the first scenario $M_{min} = M_{max}$ and thus no OA is employed. In the second scenario $M_{min} \neq M_{max}$, thus OA is tested. For each data set, the following pairs of values of (M_{min}, M_{max}) are tested in OEOA:

Table 3: Results of the on-line ensemble learning algorithms using the artificial data sets.

Approach	Base model	Ensemble size	Average and SD of MSE[1] for different values of T_0					Av. and SD of MSE[1] over all values of T_0	Av. and SD of Proc. Time[2] PTAS (min.) / PTPS (sec.)
			$T_0 = 20$	$T_0 = 40$	$T_0 = 60$	$T_0 = 80$	$T_0 = 100$		
Hyperplane data set									
AddExp	OS-ELM	$M_{max} = 15$	7.23 (0.26)	6.77 (0.17)	7.16 (0.14)	7.69 (0.15)	8.10 (0.10)	7.39 (0.52)	9.84 (0.30) / 0.30 (0.01)
DOER	OS-ELM	$M_{max} = 15$	5.87 (0.11)	5.57 (0.05)	5.80 (0.06)	6.18 (0.06)	6.50 (0.05)	5.98 (0.36)	7.72 (0.69) / 0.23 (0.02)
EOS-ELM	OS-ELM	$M_{max} = 15$	19.20 (0.69)	19.61 (0.19)	19.79 (0.16)	20.03 (0.12)	20.14 (0.10)	19.75 (0.37)	0.52 (0.11) / 0.02 (0.00)
FOS-ELM	OS-ELM	$M_{max} = 15$	6.38 (0.07)	8.02 (0.04)	10.21 (0.06)	12.43 (0.07)	15.40 (0.09)	10.49 (3.57)	0.42 (0.10) / 0.01 (0.00)
OAUE	OS-ELM	$M_{max} = 15$	19.68 (0.20)	19.59 (0.12)	19.76 (0.27)	19.74 (0.17)	19.85 (0.19)	19.72 (0.10)	1.29 (0.35) / 0.04 (0.01)
OB	OS-ELM	$M_{max} = 15$	19.01 (0.26)	19.55 (0.19)	19.77 (0.20)	20.04 (0.14)	20.16 (0.14)	19.71 (0.46)	0.44 (0.13) / 0.01 (0.00)
OEOA	OS-ELM	$M_{min} = 15$, $M_{max} = 15$	**5.82 (0.07)**	**5.54 (0.07)**	5.80 (0.05)	6.16 (0.04)	**6.49 (0.06)**	**5.96 (0.37)**	7.08 (0.95) / 0.21 (0.02)
OEOA	OS-ELM	$M_{min} = 10$, $M_{max} = 15$	5.87 (0.09)	**5.54 (0.05)**	**5.76 (0.07)**	**6.15 (0.08)**	**6.49 (0.08)**	**5.96 (0.37)**	7.25 (0.83) / 0.21 (0.02)
AddExp	λ_{DFF}OS-ELM	$M_{max} = 15$	7.26 (0.23)	6.47 (0.14)	6.51 (0.16)	6.58 (0.12)	6.70 (0.14)	6.70 (0.32)	7.91 (0.69) / 0.24 (0.02)
DOER	λ_{DFF}OS-ELM	$M_{max} = 15$	6.30 (0.09)	5.51 (0.06)	5.58 (0.07)	5.77 (0.05)	6.04 (0.06)	5.84 (0.33)	8.13 (0.75) / 0.24 (0.02)
EOS-ELM	λ_{DFF}OS-ELM	$M_{max} = 15$	8.82 (1.77)	6.35 (0.28)	6.31 (0.14)	6.37 (0.10)	6.51 (0.12)	6.87 (1.09)	0.47 (0.03) / 0.01 (0.00)
OAUE	λ_{DFF}OS-ELM	$M_{max} = 15$	10.21 (0.83)	8.49 (0.59)	8.23 (0.43)	8.51 (0.51)	8.89 (0.37)	8.87 (0.78)	1.20 (0.67) / 0.04 (0.02)
OB	λ_{DFF}OS-ELM	$M_{max} = 15$	8.57 (0.73)	6.38 (0.15)	6.41 (0.12)	6.59 (0.16)	6.84 (0.12)	6.96 (0.92)	0.50 (0.01) / 0.02 (0.00)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 15$, $M_{max} = 15$	6.23 (0.10)	5.53 (0.05)	5.57 (0.08)	5.76 (0.05)	6.01 (0.07)	5.82 (0.30)	7.07 (0.86) / 0.21 (0.03)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 10$, $N_{max} = 30$	**6.01 (0.09)**	**5.44 (0.05)**	**5.50 (0.07)**	**5.67 (0.06)**	**5.88 (0.08)**	**5.70 (0.24)**	7.19 (0.76) / 0.22 (0.02)
Learn^{++}.NSE	ELM	$M_{max} = 15$	23.63 (1.17)	12.92 (1.35)	11.47 (0.49)	10.33 (0.43)	11.97 (0.40)	14.06 (5.43)	0.54 (0.09) / 0.02 (0.00)
OWE	ELM	$M_{max} = 15$	**7.91 (0.19)**	**6.32 (0.08)**	**6.48 (0.08)**	**6.87 (0.08)**	**7.29 (0.10)**	**6.97 (0.64)**	9.00 (0.75) / 0.27 (0.02)
Friedman-LA data set									
AddExp	OS-ELM	$M_{max} = 15$	7.23 (0.16)	6.71 (0.12)	6.66 (0.09)	6.60 (0.08)	6.59 (0.07)	6.76 (0.27)	5.99 (0.47) / 0.18 (0.01)
DOER	OS-ELM	$M_{max} = 15$	7.29 (0.11)	6.75 (0.06)	6.75 (0.05)	6.72 (0.05)	6.67 (0.06)	6.84 (0.26)	6.29 (0.36) / 0.19 (0.01)
EOS-ELM	OS-ELM	$M_{max} = 15$	8.98 (1.87)	6.53 (0.14)	6.49 (0.13)	**6.45 (0.14)**	**6.42 (0.10)**	6.97 (1.12)	0.55 (0.01 / 0.02 (0.00)
FOS-ELM	OS-ELM	$M_{max} = 15$	7.27 (0.08)	6.73 (0.08)	6.51 (0.07)	6.56 (0.06)	6.52 (0.07)	6.72 (0.32)	0.42 (0.10) / 0.01 (0.00)
OAUE	OS-ELM	$M_{max} = 15$	**6.62 (0.14)**	**6.48 (0.09)**	6.53 (0.10)	6.64 (0.15)	6.56 (0.13)	6.56 (0.07)	1.60 (0.94) / 0.05 (0.03)
OB	OS-ELM	$M_{max} = 15$	8.70 (0.86)	6.60 (0.15)	6.45 (0.08)	6.46 (0.08)	6.43 (0.07)	6.93 (0.99)	0.71 (0.03) / 0.02 (0.00)
OEOA	OS-ELM	$M_{min} = 15$, $M_{max} = 15$	7.22 (0.12)	6.72 (0.06)	6.76 (0.05)	6.70 (0.05)	6.65 (0.04)	6.81 (0.23)	5.55 (0.36) / 0.16 (0.01)
OEOA	OS-ELM	$M_{min} = 10$, $M_{max} = 30$	6.86 (0.09)	**6.48 (0.04)**	**6.48 (0.07)**	6.47 (0.05)	6.46 (0.05)	**6.55 (0.17)**	6.48 (0.43) / 0.19 (0.01)
AddExp	λ_{DFF}OS-ELM	$M_{max} = 15$	8.84 (0.17)	7.99 (0.13)	7.70 (0.09)	7.49 (0.10)	7.37 (0.07)	7.88 (0.59)	7.13 (0.50) / 0.21 (0.02)
DOER	λ_{DFF}OS-ELM	$M_{max} = 15$	8.79 (0.12)	7.51 (0.08)	7.18 (0.06)	**6.94 (0.05)**	**6.80 (0.06)**	7.45 (0.80)	5.95 (1.52) / 0.18 (0.05)
EOS-ELM	λ_{DFF}OS-ELM	$M_{max} = 15$	11.34 (2.25)	8.12 (0.34)	7.73 (0.13)	7.64 (0.15)	7.55 (0.11)	8.48 (1.61)	0.45 (0.01) / 0.01 (0.00)
OAUE	λ_{DFF}OS-ELM	$M_{max} = 15$	**6.64 (0.15)**	**6.62 (0.10)**	**6.77 (0.09)**	**6.94 (0.09)**	7.10 (0.16)	**6.81 (0.21)**	1.23 (0.70) / 0.04 (0.02)
OB	λ_{DFF}OS-ELM	$M_{max} = 15$	9.48 (0.68)	7.45 (0.12)	7.27 (0.15)	7.20 (0.11)	7.11 (0.09)	7.70 (1.00)	0.48 (0.01) / 0.01 (0.00)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 15$, $M_{max} = 15$	8.73 (0.11)	7.53 (0.06)	7.17 (0.06)	6.96 (0.04)	**6.80 (0.05)**	7.44 (0.77)	5.99 (0.27) / 0.18 (0.01)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 10$, $M_{max} = 30$	8.40 (0.08)	7.48 (0.06)	7.15 (0.06)	6.95 (0.05)	6.81 (0.05)	7.36 (0.63)	6.65 (0.41) / 0.20 (0.01)
Learn^{++}.NSE	ELM	$M_{max} = 15$	20.45 (0.93)	13.30 (0.53)	10.79 (0.39)	9.57 (0.37)	8.93 (0.25)	12.61 (4.69)	0.71 (0.28) / 0.02 (0.01)
OWE	ELM	$M_{max} = 15$	**9.26 (0.14)**	**7.41 (0.07)**	**7.10 (0.05)**	**6.89 (0.06)**	**6.76 (0.06)**	**7.49 (1.02)**	6.62 (0.78) / 0.20 (0.02)
Friedman-GRA data set									
AddExp	OS-ELM	$M_{max} = 15$	12.40 (0.29)	11.55 (0.16)	11.51 (0.13)	11.46 (0.12)	11.46 (0.12)	11.67 (0.41)	8.78 (1.29) / 0.26 (0.04)
DOER	OS-ELM	$M_{max} = 15$	12.63 (0.16)	11.43 (0.10)	11.30 (0.12)	11.17 (0.08)	11.08 (0.06)	11.52 (0.63)	9.40 (0.97) / 0.28 (0.03)
EOS-ELM	OS-ELM	$M_{max} = 15$	16.30 (3.18)	12.42 (0.64)	11.92 (0.11)	11.84 (0.13)	11.85 (0.11)	12.87 (1.94)	0.55 (0.01) / 0.02 (0.00)
FOS-ELM	OS-ELM	$M_{max} = 15$	**11.88 (0.17)**	**11.16 (0.10)**	11.32 (0.12)	11.53 (0.08)	12.00 (0.09)	11.58 (0.36)	0.43 (0.11) / 0.01 (0.00)
OAUE	OS-ELM	$M_{max} = 15$	12.45 (0.22)	12.29 (0.28)	12.16 (0.14)	12.33 (0.30)	12.28 (0.29)	12.30 (0.10)	1.66 (0.97) / 0.05 (0.03)
OB	OS-ELM	$M_{max} = 15$	14.83 (1.48)	12.26 (0.31)	11.99 (0.11)	11.94 (0.11)	11.88 (0.10)	12.58 (1.27)	0.74 (0.13) / 0.02 (0.00)
OEOA	OS-ELM	$M_{min} = 15$, $M_{max} = 15$	12.60 (0.16)	11.40 (0.10)	11.33 (0.08)	11.14 (0.09)	11.04 (0.07)	11.50 (0.63)	8.92 (1.11) / 0.26 (0.03)
OEOA	OS-ELM	$M_{min} = 10$, $M_{max} = 30$	11.96 (0.13)	**11.16 (0.11)**	**11.02 (0.05)**	**10.97 (0.09)**	**10.89 (0.08)**	**11.20 (0.43)**	9.63 (0.70) / 0.28 (0.02)
AddExp	λ_{DFF}OS-ELM	$M_{max} = 15$	14.24 (0.38)	12.60 (0.19)	12.15 (0.13)	11.87 (0.18)	11.63 (0.12)	12.50 (1.04)	8.63 (0.38) / 0.26 (0.01)
DOER	λ_{DFF}OS-ELM	$M_{max} = 15$	14.47 (0.23)	12.21 (0.13)	11.61 (0.11)	11.27 (0.08)	11.09 (0.10)	12.13 (1.38)	7.74 (0.30) / 0.23 (0.01)
EOS-ELM	λ_{DFF}OS-ELM	$M_{max} = 15$	16.03 (1.52)	13.69 (1.21)	12.46 (0.42)	12.50 (0.37)	12.16 (0.26)	13.37 (1.60)	0.45 (0.01) / 0.01 (0.00)
OAUE	λ_{DFF}OS-ELM	$M_{max} = 15$	**12.23 (0.31)**	**11.90 (0.42)**	11.88 (0.19)	11.90 (0.25)	11.97 (0.41)	11.98 (0.15)	1.25 (0.79) / 0.04 (0.02)
OB	λ_{DFF}OS-ELM	$M_{max} = 15$	15.09 (0.82)	12.20 (0.50)	11.74 (0.21)	11.53 (0.25)	11.44 (0.17)	12.40 (1.53)	0.48 (0.01) / 0.01 (0.00)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 15$, $M_{max} = 15$	14.51 (0.16)	12.16 (0.10)	11.62 (0.10)	11.27 (0.10)	**11.04 (0.08)**	12.12 (1.40)	8.81 (1.19) / 0.26 (0.04)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 10$, $M_{max} = 30$	13.66 (0.15)	12.07 (0.10)	**11.57 (0.10)**	**11.17 (0.08)**	**11.04 (0.09)**	**11.90 (1.06)**	8.87 (0.40) / 0.27 (0.01)
Learn^{++}.NSE	ELM	$M_{max} = 15$	37.56 (10.02)	21.34 (0.93)	17.81 (0.64)	16.06 (0.55)	16.03 (0.69)	21.76 (9.09)	0.68 (0.29) / 0.02 (0.01)
OWE	ELM	$M_{max} = 15$	**14.54 (0.33)**	**12.10 (0.17)**	**11.59 (0.13)**	**11.29 (0.10)**	**11.21 (0.08)**	**12.15 (1.38)**	7.89 (2.19) / 0.24 (0.07)
Friedman-GnRG data set									
AddExp	OS-ELM	$M_{max} = 15$	11.63 (0.19)	10.87 (0.12)	10.70 (0.09)	10.67 (0.11)	10.62 (0.07)	10.90 (0.42)	7.54 (1.25) / 0.23 (0.04)
DOER	OS-ELM	$M_{max} = 15$	11.54 (0.16)	10.51 (0.09)	10.27 (0.06)	10.15 (0.06)	9.98 (0.07)	10.49 (0.61)	8.90 (0.85) / 0.27 (0.03)
EOS-ELM	OS-ELM	$M_{max} = 15$	13.73 (1.53)	12.40 (0.44)	12.27 (0.13)	12.27 (0.15)	12.20 (0.10)	12.57 (0.65)	0.56 (0.00) / 0.02 (0.00)
FOS-ELM	OS-ELM	$M_{max} = 15$	**10.57 (0.11)**	**10.19 (0.09)**	10.21 (0.11)	10.66 (0.09)	11.02 (0.06)	10.53 (0.35)	0.41 (0.12) / 0.01 (0.00)
OAUE	OS-ELM	$M_{max} = 15$	12.54 (0.14)	12.45 (0.17)	12.53 (0.19)	12.66 (0.26)	12.71 (0.45)	12.58 (0.11)	1.37 (1.10) / 0.04 (0.03)
OB	OS-ELM	$M_{max} = 15$	13.70 (0.52)	12.30 (0.19)	12.22 (0.10)	12.25 (0.10)	12.27 (0.12)	12.55 (0.65)	0.71 (0.05) / 0.02 (0.00)
OEOA	OS-ELM	$M_{min} = 15$, $M_{max} = 15$	11.51 (0.17)	10.53 (0.09)	10.26 (0.11)	10.14 (0.11)	9.98 (0.06)	10.48 (0.61)	8.15 (1.16) / 0.24 (0.03)
OEOA	OS-ELM	$M_{min} = 10$, $M_{max} = 30$	10.99 (0.13)	10.23 (0.06)	**10.05 (0.06)**	**9.97 (0.06)**	**9.84 (0.06)**	**10.22 (0.45)**	9.04 (0.57) / 0.27 (0.01)
AddExp	λ_{DFF}OS-ELM	$M_{max} = 15$	13.02 (0.21)	11.60 (0.18)	11.12 (0.14)	10.81 (0.12)	10.59 (0.11)	11.43 (0.97)	8.39 (0.69) / 0.25 (0.02)
DOER	λ_{DFF}OS-ELM	$M_{max} = 15$	13.25 (0.17)	11.20 (0.07)	10.56 (0.10)	10.28 (0.08)	**10.03 (0.07)**	11.06 (1.30)	8.31 (0.53) / 0.25 (0.02)
EOS-ELM	λ_{DFF}OS-ELM	$M_{max} = 15$	14.63 (2.59)	12.07 (0.74)	11.65 (0.35)	11.40 (0.43)	11.09 (0.25)	12.17 (1.42)	0.46 (0.01) / 0.01 (0.00)
OAUE	λ_{DFF}OS-ELM	$M_{max} = 15$	**11.97 (0.22)**	11.60 (0.21)	11.49 (0.30)	11.30 (0.31)	11.21 (0.30)	11.52 (0.30)	1.22 (0.53) / 0.04 (0.02)
OB	λ_{DFF}OS-ELM	$M_{max} = 15$	13.72 (1.00)	11.06 (0.26)	10.77 (0.18)	10.59 (0.10)	10.52 (0.17)	11.33 (1.35)	0.49 (0.01) / 0.01 (0.00)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 15$, $M_{max} = 15$	13.15 (0.13)	11.21 (0.10)	10.58 (0.09)	**10.26 (0.07)**	10.04 (0.07)	11.05 (1.26)	8.44 (1.36) / 0.25 (0.04)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 10$, $M_{max} = 30$	12.46 (0.14)	**11.04 (0.09)**	**10.50 (0.09)**	**10.26 (0.07)**	10.05 (0.05)	**10.86 (0.97)**	8.55 (0.60) / 0.26 (0.02)
Learn^{++}.NSE	ELM	$M_{max} = 15$	29.73 (1.03)	19.42 (0.84)	15.50 (0.57)	14.16 (0.44)	13.34 (0.58)	18.43 (6.73)	0.68 (0.31) / 0.02 (0.01)
OWE	ELM	$M_{max} = 15$	**14.44 (0.26)**	**11.21 (0.14)**	**10.52 (0.07)**	**10.30 (0.09)**	**10.06 (0.09)**	**11.31 (1.80)**	9.80 (1.12) / 0.29 (0.03)

[1]The MSE values have been multiplied by 10^3; [2]Av. and SD of the Processing Time on all samples (PTAS) and per sample (PTPS) over all values of T_0.

Table 4: Results of the on-line ensemble learning algorithms using the real-world data sets of small size.

Approach	Base model	Ensemble size	Average and SD of MSE[1] for different values of T_0					Av. and SD of MSE[1] over all values of T_0	Av. and SD of Proc. Time[2] PTAS (min.) / PTPS (sec.)
			$T_0 = 10$	$T_0 = 20$	$T_0 = 30$	$T_0 = 40$	$T_0 = 50$		
Polymerization reactor data set									
AddExp	OS-ELM	$M_{max} = 15$	2.80 (0.09)	2.73 (0.14)	2.67 (0.20)	2.52 (0.17)	2.75 (0.21)	2.70 (0.11)	1.28 (0.35) / 0.12 (0.03)
DOER	OS-ELM	$M_{max} = 15$	0.47 (0.04)	0.66 (0.06)	0.80 (0.07)	0.99 (0.08)	1.22 (0.09)	0.83 (0.29)	1.41 (0.51) / 0.13 (0.05)
EOS-ELM	OS-ELM	$M_{max} = 15$	8.09 (0.42)	14.18 (3.07)	6.01 (0.25)	4.54 (0.26)	4.12 (0.20)	7.39 (4.10)	0.26 (0.04) / 0.02 (0.00)
FOS-ELM	OS-ELM	$M_{max} = 15$	3.91 (0.27)	5.22 (0.39)	4.12 (0.23)	3.94 (0.25)	3.00 (0.28)	4.04 (0.79)	0.21 (0.06) / 0.02 (0.01)
OAUE	OS-ELM	$M_{max} = 15$	2.76 (0.23)	2.93 (0.19)	3.59 (0.34)	3.74 (0.42)	4.06 (0.37)	3.42 (0.55)	0.17 (0.04) / 0.02 (0.00)
OB	OS-ELM	$M_{max} = 15$	8.32 (0.57)	10.73 (1.47)	5.99 (0.34)	4.49 (0.25)	4.27 (0.24)	6.76 (2.75)	0.28 (0.03) / 0.03 (0.00)
OEOA	OS-ELM	$M_{min} = 15$, $M_{max} = 15$	0.50 (0.02)	0.68 (0.05)	0.81 (0.07)	1.10 (0.12)	1.37 (0.08)	0.89 (0.35)	1.36 (0.45) / 0.12 (0.04)
OEOA	OS-ELM	$M_{min} = 5$, $M_{max} = 30$	**0.40 (0.06)**	**0.52 (0.06)**	**0.69 (0.09)**	**0.87 (0.10)**	**1.12 (0.09)**	**0.72 (0.28)**	1.38 (0.36) / 0.12 (0.03)
AddExp	λ_{DFF}OS-ELM	$M_{max} = 15$	0.97 (0.12)	1.33 (0.19)	1.47 (0.16)	1.58 (0.25)	1.62 (0.15)	1.39 (0.26)	0.31 (0.06) / 0.03 (0.01)
DOER	λ_{DFF}OS-ELM	$M_{max} = 15$	**0.29 (0.02)**	0.46 (0.04)	0.54 (0.05)	0.64 (0.09)	0.88 (0.05)	0.56 (0.22)	0.88 (0.29) / 0.08 (0.03)
EOS-ELM	λ_{DFF}OS-ELM	$M_{max} = 15$	0.68 (0.13)	2.94 (2.42)	1.50 (0.38)	2.23 (0.26)	2.25 (0.21)	1.92 (0.86)	0.22 (0.04) / 0.02 (0.00)
OAUE	λ_{DFF}OS-ELM	$M_{max} = 15$	0.81 (0.13)	1.45 (0.40)	1.76 (0.47)	2.31 (0.34)	2.40 (0.51)	1.75 (0.65)	0.40 (0.14) / 0.04 (0.01)
OB	λ_{DFF}OS-ELM	$M_{max} = 15$	1.03 (0.16)	2.13 (0.86)	2.13 (0.27)	2.75 (0.26)	2.61 (0.17)	2.13 (0.68)	0.21 (0.03) / 0.02 (0.00)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 15$, $M_{max} = 15$	**0.29 (0.03)**	0.48 (0.03)	0.52 (0.05)	0.62 (0.05)	0.87 (0.06)	0.56 (0.21)	0.94 (0.32) / 0.09 (0.03)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 5$, $M_{max} = 30$	0.30 (0.03)	**0.43 (0.05)**	**0.51 (0.04)**	**0.60 (0.07)**	**0.82 (0.09)**	**0.53 (0.19)**	1.25 (0.30) / 0.12 (0.03)
Learn^{++}.NSE	ELM	$M_{max} = 15$	2.89 (0.42)	5.74 (0.87)	6.74 (1.30)	8.05 (2.69)	16.83 (2.99)	8.05 (5.26)	0.26 (0.09) / 0.02 (0.01)
OWE	ELM	$M_{max} = 15$	**0.55 (0.06)**	**0.94 (0.10)**	**1.33 (0.11)**	**1.90 (0.12)**	**2.42 (0.27)**	**1.43 (0.75)**	1.85 (0.81) / 0.17 (0.08)
Cement kiln data set									
AddExp	OS-ELM	$M_{max} = 15$	12.13 (0.26)	12.22 (0.23)	12.02 (0.35)	11.97 (0.24)	11.86 (0.34)	12.04 (0.14)	3.39 (1.10) / 0.29 (0.09)
DOER	OS-ELM	$M_{max} = 15$	10.03 (0.19)	10.25 (0.21)	10.20 (0.23)	10.26 (0.20)	10.13 (0.18)	10.17 (0.10)	3.17 (1.20) / 0.27 (0.10)
EOS-ELM	OS-ELM	$M_{max} = 15$	22.40 (1.17)	22.05 (2.00)	17.33 (2.31)	18.59 (2.64)	17.88 (1.98)	19.65 (2.39)	0.28 (0.05) / 0.02 (0.00)
FOS-ELM	OS-ELM	$M_{max} = 15$	17.70 (5.59)	18.91 (8.29)	20.12 (0.69)	23.53 (0.97)	24.16 (1.39)	20.89 (2.84)	0.30 (0.09) / 0.03 (0.01)
OAUE	OS-ELM	$M_{max} = 15$	12.07 (0.32)	12.67 (0.32)	12.97 (0.40)	13.47 (0.64)	14.99 (1.15)	13.23 (1.11)	0.49 (0.10) / 0.04 (0.01)
OB	OS-ELM	$M_{max} = 15$	20.08 (0.86)	18.70 (1.09)	16.52 (1.12)	16.46 (0.96)	16.13 (0.91)	17.58 (1.73)	0.31 (0.05) / 0.03 (0.00)
OEOA	OS-ELM	$M_{min} = 15$, $M_{max} = 15$	9.96 (0.18)	10.22 (0.14)	10.08 (0.23)	10.21 (0.20)	10.15 (0.23)	10.12 (0.11)	2.85 (1.10) / 0.24 (0.09)
OEOA	OS-ELM	$M_{min} = 5$, $M_{max} = 30$	**9.26 (0.17)**	**9.59 (0.17)**	**9.56 (0.29)**	**9.73 (0.25)**	**9.69 (0.22)**	**9.56 (0.18)**	3.18 (1.16) / 0.27 (0.10)
AddExp	λ_{DFF}OS-ELM	$M_{max} = 15$	9.13 (0.24)	9.44 (0.25)	9.35 (0.23)	9.67 (0.32)	9.78 (0.31)	9.47 (0.26)	2.47 (0.72) / 0.21 (0.06)
DOER	λ_{DFF}OS-ELM	$M_{max} = 15$	8.61 (0.11)	8.86 (0.11)	8.78 (0.15)	8.97 (0.16)	8.97 (0.20)	8.84 (0.15)	2.68 (0.97) / 0.23 (0.08)
EOS-ELM	λ_{DFF}OS-ELM	$M_{max} = 15$	10.03 (0.63)	10.20 (1.33)	9.15 (1.05)	9.49 (0.91)	9.46 (1.14)	9.67 (0.44)	0.24 (0.05) / 0.02 (0.00)
OAUE	λ_{DFF}OS-ELM	$M_{max} = 15$	9.61 (0.30)	9.55 (0.28)	9.45 (0.35)	9.70 (0.56)	10.17 (0.81)	9.70 (0.28)	0.57 (0.20) / 0.05 (0.02)
OB	λ_{DFF}OS-ELM	$M_{max} = 15$	9.44 (0.43)	9.36 (0.34)	8.94 (0.21)	9.19 (0.28)	9.16 (0.20)	9.22 (0.19)	0.23 (0.04) / 0.02 (0.00)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 15$, $M_{max} = 15$	8.60 (0.09)	8.88 (0.13)	8.77 (0.15)	8.93 (0.15)	8.92 (0.19)	8.82 (0.14)	2.70 (1.05) / 0.23 (0.09)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 10$, $M_{max} = 30$	**8.57 (0.09)**	**8.71 (0.13)**	**8.67 (0.08)**	**8.81 (0.08)**	**8.79 (0.11)**	**8.71 (0.10)**	4.40 (2.02) / 0.38 (0.17)
Learn^{++}.NSE	ELM	$M_{max} = 15$	23.12 (3.03)	24.47 (1.71)	28.51 (2.03)	26.75 (3.51)	33.44 (5.59)	27.26 (4.03)	0.31 (0.07) / 0.03 (0.01)
OWE	ELM	$M_{max} = 15$	**11.23 (0.41)**	**12.67 (0.69)**	**13.04 (0.67)**	**13.55 (0.65)**	**13.16 (0.49)**	**12.73 (0.89)**	3.40 (1.63) / 0.29 (0.14)

[1]The MSE values have been multiplied by 10^3; [2]Av. and SD of the Processing Time on all samples (PTAS) and per sample (PTPS) over all values of T_0.

$(5, 15)$, $(5, 30)$, $(10, 15)$, and $(10, 30)$; and for each data set and base model, and for all the tested values of T_0, the pair with the lowest average MSE error is presented in the in the results of the experiments below. The simulation results are presented in Tables 3, 4, and 5. Several values of T_0 were tested, as described in these tables. For each problem, the simulation was conducted 20 times. The average and standard deviation values of the MSE, and of the processing time on all samples (PTAS) in minutes, and processing time per sample (PTPS) in seconds, with respect to all values of T_0 are presented. The processing time considers the time spent on both the training and on-line phases.

As described in Section 5.4, λ_{DFF}OS-ELM has poor performance when compared to OS-ELM in the Friedman data sets. This can be observed in Table 3 where most ensembles have better performance when OS-ELM is the base model. The exceptions are for the OAUE in the Friedman-GRA and Friedman-GnRG data sets; and for the OB in the Friedman-GnRG data set; where the error is reduced when λ_{DFF}OS-ELM is the base model. For the other data sets, in most cases, the ensembles' errors reduce significantly when λ_{DFF}OS-ELM is the base model. The reduction can be observed mainly in the debutanizer column data set. For example, for the EOS-ELM with OS-ELM as base model, the average of the MSE over all tested values of T_0 is 21.3×10^{-3}; and with λ_{DFF}OS-ELM as base model, the average of the MSE is reduced to 2.34×10^{-3}; And for the OB with OS-ELM as base model, the average of MSE over all values of T_0 is 21.11×10^{-3}; and with λ_{DFF}OS-ELM as base model, the MSE is reduced to 3.47×10^{-3}. OB and EOS-ELM are ensembles with few adaptive mechanisms, since only retraining of models is employed, and no weights' adaptation and no dynamic selection of models are employed. However, they significantly improve their performances when λ_{DFF}OS-ELM is the base model. Additionally, they have low processing time when compared to the other approaches.

OWE outperforms Learn^{++}.NSE in all cases, and FOS-ELM in most cases (except in the Friedman data sets). This is because, Learn^{++}.NSE and FOS-ELM are adapted on a batch basis; while OWE is adapted on a sample basis. Therefore, OWE adapts faster to changes. The best performances of OWE are achieved mainly in the hyperplane data set and the thermal oxidizer data set. In the thermal oxidizer data set, the average of the MSE over all values of T_0 for the OWE is 1.17×10^{-3}; while for OEOA with ordering ($M_{min} \neq M_{max}$) and λ_{DFF}OS-ELM as base model, the average MSE is 1.18×10^{-3}. In contrast to OWE and Learn^{++}.NSE, OAUE retrains all the models at each new sample. However, OAUE includes new models into the ensemble at a low frequency when compared to AddExp, OEOA,

Table 5: Results of the on-line ensemble learning algorithms the real-world data sets of medium size.

Approach	Base model	Ensemble size	Average and SD of MSE[1] for different values of T_0					Av. and SD of MSE[1] over all values of T_0	Av. and SD of Proc. Time[2] PTAS (min.) / PTPS (sec.)
			$T_0 = 20$	$T_0 = 40$	$T_0 = 60$	$T_0 = 80$	$T_0 = 100$		
Powder detergent data set									
AddExp	OS-ELM	$M_{max} = 15$	5.13 (0.09)	5.11 (0.10)	5.21 (0.09)	5.29 (0.11)	5.37 (0.06)	5.22 (0.11)	5.14 (1.11) / 0.16 (0.03)
DOER	OS-ELM	$M_{max} = 15$	4.53 (0.06)	4.76 (0.09)	4.97 (0.08)	5.09 (0.08)	5.19 (0.06)	4.91 (0.27)	7.47 (0.22) / 0.23 (0.01)
EOS-ELM	OS-ELM	$M_{max} = 15$	6.92 (0.70)	6.04 (0.47)	6.14 (0.37)	5.94 (0.17)	5.83 (0.22)	6.17 (0.43)	0.54 (0.01) / 0.02 (0.00)
FOS-ELM	OS-ELM	$M_{max} = 15$	7.76 (0.44)	8.47 (0.30)	8.05 (0.22)	6.33 (0.14)	5.63 (0.06)	7.25 (1.21)	0.41 (0.14) / 0.01 (0.00)
OAUE	OS-ELM	$M_{max} = 15$	5.09 (0.07)	5.36 (0.11)	5.59 (0.17)	5.72 (0.16)	5.79 (0.20)	5.51 (0.29)	1.36 (0.79) / 0.04 (0.02)
OB	OS-ELM	$M_{max} = 15$	6.43 (0.26)	5.87 (0.15)	5.80 (0.15)	5.76 (0.12)	5.72 (0.08)	5.92 (0.29)	0.68 (0.05) / 0.02 (0.00)
OEOA	OS-ELM	$M_{min} = 15$, $M_{max} = 15$	4.51 (0.06)	4.74 (0.08)	4.95 (0.10)	5.07 (0.09)	5.17 (0.11)	4.89 (0.27)	6.54 (0.42) / 0.20 (0.01)
OEOA	OS-ELM	$M_{min} = 5$, $M_{max} = 30$	**4.31 (0.09)**	**4.51 (0.11)**	**4.69 (0.11)**	**4.84 (0.13)**	**4.96 (0.11)**	**4.66 (0.26)**	7.66 (0.52) / 0.23 (0.01)
AddExp	λ_{DFF}OS-ELM	$M_{max} = 15$	4.07 (0.15)	4.05 (0.10)	4.15 (0.12)	4.20 (0.08)	4.26 (0.10)	4.15 (0.09)	4.48 (0.68) / 0.14 (0.02)
DOER	λ_{DFF}OS-ELM	$M_{max} = 15$	4.00 (0.04)	4.17 (0.07)	4.18 (0.08)	4.27 (0.07)	4.40 (0.08)	4.20 (0.15)	7.02 (2.22) / 0.21 (0.07)
EOS-ELM	λ_{DFF}OS-ELM	$M_{max} = 15$	3.86 (0.14)	**3.78 (0.08)**	**3.75 (0.05)**	3.86 (0.07)	**3.84 (0.06)**	**3.82 (0.05)**	0.45 (0.01) / 0.01 (0.00)
OAUE	λ_{DFF}OS-ELM	$M_{max} = 15$	3.89 (0.08)	3.91 (0.10)	3.96 (0.13)	4.20 (0.19)	4.21 (0.17)	4.03 (0.16)	1.45 (1.00) / 0.04 (0.03)
OB	λ_{DFF}OS-ELM	$M_{max} = 15$	**3.84 (0.08)**	3.79 (0.06)	3.80 (0.09)	**3.84 (0.08)**	3.89 (0.06)	3.83 (0.04)	0.46 (0.00) / 0.01 (0.00)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 15$, $M_{max} = 15$	3.99 (0.07)	4.11 (0.10)	4.19 (0.11)	4.26 (0.09)	4.35 (0.10)	4.18 (0.14)	6.23 (0.27) / 0.19 (0.01)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 10$, $M_{max} = 30$	3.87 (0.05)	3.87 (0.06)	3.90 (0.06)	3.99 (0.06)	4.10 (0.08)	3.95 (0.10)	6.54 (0.56) / 0.20 (0.02)
Learn^{++}.NSE	ELM	$M_{max} = 15$	9.94 (1.14)	11.51 (1.66)	12.01 (1.68)	11.77 (2.47)	11.07 (1.69)	11.26 (0.82)	0.64 (0.28) / 0.02 (0.01)
OWE	ELM	$M_{max} = 15$	**5.88 (0.17)**	**6.28 (0.21)**	**6.67 (0.20)**	**7.06 (0.20)**	**7.12 (0.18)**	**6.60 (0.53)**	7.36 (1.57) / 0.23 (0.05)
Thermal oxidizer data set									
AddExp	OS-ELM	$M_{max} = 15$	1.45 (0.04)	1.40 (0.04)	1.38 (0.03)	1.38 (0.03)	1.39 (0.03)	1.40 (0.03)	1.53 (0.26) / 0.04 (0.01)
DOER	OS-ELM	$M_{max} = 15$	**1.12 (0.01)**	1.13 (0.01)	1.12 (0.01)	1.13 (0.01)	1.13 (0.00)	1.13 (0.01)	7.41 (2.24) / 0.22 (0.07)
EOS-ELM	OS-ELM	$M_{max} = 15$	1.66 (0.10)	1.64 (0.13)	1.79 (0.08)	1.78 (0.13)	1.70 (0.12)	1.72 (0.07)	0.66 (0.07) / 0.02 (0.00)
FOS-ELM	OS-ELM	$M_{max} = 15$	1.41 (0.09)	1.46 (0.08)	1.33 (0.05)	1.45 (0.10)	1.55 (0.11)	1.44 (0.08)	0.45 (0.10) / 0.01 (0.00)
OAUE	OS-ELM	$M_{max} = 15$	1.19 (0.01)	1.26 (0.01)	1.29 (0.02)	1.34 (0.03)	1.39 (0.03)	1.30 (0.07)	1.64 (0.97) / 0.05 (0.03)
OB	OS-ELM	$M_{max} = 15$	1.64 (0.03)	1.57 (0.05)	1.65 (0.04)	1.70 (0.05)	1.66 (0.04)	1.64 (0.04)	0.79 (0.07) / 0.02 (0.00)
OEOA	OS-ELM	$M_{min} = 15$, $M_{max} = 15$	**1.12 (0.01)**	1.13 (0.01)	1.12 (0.01)	1.13 (0.00)	1.14 (0.00)	1.13 (0.01)	7.41 (2.41) / 0.21 (0.07)
OEOA	OS-ELM	$M_{min} = 5$, $M_{max} = 15$	**1.12 (0.01)**	**1.12 (0.01)**	**1.11 (0.01)**	**1.11 (0.01)**	**1.12 (0.01)**	**1.12 (0.01)**	6.54 (2.17) / 0.19 (0.06)
AddExp	λ_{DFF}OS-ELM	$M_{max} = 15$	1.32 (0.04)	1.28 (0.05)	1.27 (0.05)	1.28 (0.04)	1.26 (0.05)	1.28 (0.02)	0.80 (0.16) / 0.02 (0.00)
DOER	λ_{DFF}OS-ELM	$M_{max} = 15$	**1.17 (0.02)**	**1.19 (0.02)**	1.18 (0.02)	**1.17 (0.02)**	1.18 (0.02)	**1.18 (0.01)**	6.29 (1.80) / 0.18 (0.05)
EOS-ELM	λ_{DFF}OS-ELM	$M_{max} = 15$	1.26 (0.04)	1.27 (0.04)	1.24 (0.05)	1.23 (0.05)	1.27 (0.09)	1.25 (0.02)	0.54 (0.08) / 0.02 (0.00)
OAUE	λ_{DFF}OS-ELM	$M_{max} = 15$	**1.17 (0.03)**	1.22 (0.04)	1.24 (0.07)	1.27 (0.07)	1.27 (0.07)	1.23 (0.04)	1.64 (0.86) / 0.05 (0.02)
OB	λ_{DFF}OS-ELM	$M_{max} = 15$	1.22 (0.04)	1.20 (0.03)	1.19 (0.03)	1.19 (0.04)	1.20 (0.03)	1.20 (0.01)	0.57 (0.05) / 0.02 (0.00)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 15$, $M_{max} = 15$	1.18 (0.03)	**1.19 (0.01)**	**1.17 (0.02)**	1.18 (0.02)	**1.17 (0.02)**	**1.18 (0.01)**	6.73 (2.74) / 0.20 (0.08)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 10$, $M_{max} = 15$	1.18 (0.02)	1.20 (0.02)	1.18 (0.02)	1.18 (0.02)	**1.17 (0.01)**	**1.18 (0.01)**	8.48 (3.09) / 0.25 (0.09)
Learn^{++}.NSE	ELM	$M_{max} = 15$	1.36 (0.10)	1.50 (0.13)	1.59 (0.09)	1.69 (0.14)	1.76 (0.15)	1.58 (0.16)	0.83 (0.26) / 0.02 (0.01)
OWE	ELM	$M_{max} = 15$	**1.12 (0.01)**	**1.16 (0.01)**	**1.17 (0.01)**	**1.19 (0.01)**	**1.21 (0.01)**	**1.17 (0.03)**	6.88 (2.65) / 0.20 (0.08)
Debutanizer column data set									
AddExp	OS-ELM	$M_{max} = 15$	7.28 (0.19)	7.94 (0.20)	8.87 (0.15)	9.59 (0.15)	10.14 (0.18)	8.77 (1.17)	8.62 (1.10) / 0.22 (0.03)
DOER	OS-ELM	$M_{max} = 15$	2.61 (0.21)	3.62 (0.12)	5.72 (0.18)	6.73 (0.20)	7.73 (0.22)	5.28 (2.13)	14.02 (1.82) / 0.35 (0.05)
EOS-ELM	OS-ELM	$M_{max} = 15$	23.22 (0.82)	21.41 (0.54)	20.59 (0.48)	20.86 (0.45)	20.45 (0.25)	21.30 (1.13)	0.58 (0.03) / 0.01 (0.00)
FOS-ELM	OS-ELM	$M_{max} = 15$	36.34 (13.99)	170.07 (67.27)	37.34 (4.83)	36.58 (2.34)	33.73 (1.10)	62.81 (59.97)	0.43 (0.13) / 0.01 (0.00)
OAUE	OS-ELM	$M_{max} = 15$	15.22 (0.30)	17.86 (0.31)	18.72 (0.29)	19.04 (0.26)	19.30 (0.23)	18.03 (1.66)	1.81 (1.06) / 0.05 (0.03)
OB	OS-ELM	$M_{max} = 15$	22.61 (0.45)	21.25 (0.19)	20.58 (0.26)	20.56 (0.25)	20.57 (0.25)	21.11 (0.88)	0.74 (0.03) / 0.02 (0.00)
OEOA	OS-ELM	$M_{min} = 15$, $M_{max} = 15$	2.60 (0.20)	3.60 (0.11)	5.68 (0.17)	6.73 (0.16)	7.77 (0.17)	5.28 (2.15)	12.59 (1.63) / 0.31 (0.04)
OEOA	OS-ELM	$M_{min} = 5$, $M_{max} = 30$	**2.17 (0.20)**	**3.07 (0.15)**	**4.92 (0.17)**	**5.87 (0.19)**	**6.90 (0.27)**	**4.59 (1.95)**	13.12 (3.95) / 0.33 (0.10)
AddExp	λ_{DFF}OS-ELM	$M_{max} = 15$	1.39 (0.24)	1.94 (0.26)	2.66 (0.35)	**2.77 (0.31)**	3.26 (0.36)	2.40 (0.73)	3.68 (0.87) / 0.09 (0.02)
DOER	λ_{DFF}OS-ELM	$M_{max} = 15$	1.55 (0.19)	2.14 (0.10)	3.18 (0.14)	3.74 (0.15)	4.31 (0.17)	2.99 (1.14)	16.56 (1.37) / 0.42 (0.03)
EOS-ELM	λ_{DFF}OS-ELM	$M_{max} = 15$	**1.11 (0.16)**	1.89 (0.34)	2.82 (0.21)	2.84 (0.30)	**3.05 (0.16)**	2.34 (0.82)	0.50 (0.02) / 0.01 (0.00)
OAUE	λ_{DFF}OS-ELM	$M_{max} = 15$	2.19 (0.44)	2.75 (0.31)	3.31 (0.30)	3.32 (0.26)	3.47 (0.30)	3.01 (0.53)	1.68 (0.98) / 0.04 (0.02)
OB	λ_{DFF}OS-ELM	$M_{max} = 15$	2.03 (0.17)	3.09 (0.23)	3.96 (0.19)	3.95 (0.23)	4.30 (0.16)	3.47 (0.92)	0.53 (0.02) / 0.01 (0.00)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 15$, $M_{max} = 15$	1.55 (0.13)	2.15 (0.11)	3.20 (0.11)	3.72 (0.14)	4.29 (0.22)	2.98 (1.12)	8.75 (1.07) / 0.22 (0.03)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 10$, $M_{max} = 30$	1.19 (0.09)	**1.77 (0.12)**	**2.42 (0.23)**	2.84 (0.14)	3.07 (0.14)	**2.26 (0.78)**	12.13 (2.13) / 0.30 (0.05)
Learn^{++}.NSE	ELM	$M_{max} = 15$	26.49 (4.69)	42.15 (4.83)	51.06 (8.00)	72.20 (15.63)	98.71 (31.13)	58.12 (28.07)	0.77 (0.36) / 0.02 (0.01)
OWE	ELM	$M_{max} = 15$	**9.19 (0.61)**	**19.58 (1.18)**	**31.18 (1.52)**	**29.67 (1.54)**	**32.76 (1.80)**	**24.48 (9.98)**	16.79 (3.09) / 0.42 (0.08)

[1]The MSE values have been multiplied by 10^3; [2]Av. and SD of the Processing Time on all samples (PTAS) and per sample (PTPS) over all values of T_0.

OWE, and DOER. AddExp employs the same adaptive ensemble mechanisms as the OEOA. However, AddExp has an error larger than the error on OEOA in all cases. In the AddExp, new models take more time to have their weights significantly increased. In scenarios that require faster adaptation to the new concepts, this method for assigning weights may fail. In contrast to AddExp, OEOA assigns large weights to the new and accurate models if they have low errors on the newest samples.

In most cases, OEOA with OA significantly reduces the ensemble error when compared to OEOA without OA - for example, in the debutanizer column and cement kiln data sets. In other cases, OEOA with OA has similar performance when compared to OEOA without OA - for example, in the hyperplane data set with OS-ELM as the base model. In most cases, it can also be observed that OEOA with OA has better performance when M_{min} and M_{max} are large; And the sets with best performances are $(M_{min}, M_{max}) = (10, 30)$ and $(M_{min}, M_{max}) = (5, 30)$. DOER, and OEOA without OA, have similar performances, since they have similar methods for the assignment of weights and for the selection of models. However, DOER

Table 6: Results of the on-line ensemble learning algorithms on a real-world data set of large size.

Approach	Base model	Ensemble size	Average and SD of MSE[1] for different values of T_0					Av. and SD of MSE[1] over all values of T_0	Av. and SD of Proc. Time[2] PTAS (min.) / PTPS (sec.)
			$T_0 = 30$	$T_0 = 60$	$T_0 = 90$	$T_0 = 120$	$T_0 = 150$		
SRU data set									
AddExp	OS-ELM	$M_{max} = 15$	1.64 (0.02)	1.64 (0.03)	1.66 (0.03)	1.70 (0.03)	1.73 (0.03)	1.67 (0.04)	10.30 (0.69) / 0.06 (0.00)
DOER	OS-ELM	$M_{max} = 15$	1.02 (0.02)	1.17 (0.03)	1.23 (0.02)	1.27 (0.03)	1.34 (0.02)	1.20 (0.12)	19.02 (1.56) / 0.11 (0.01)
EOS-ELM	OS-ELM	$M_{max} = 15$	2.72 (0.05)	2.53 (0.05)	2.72 (0.03)	2.69 (0.03)	2.68 (0.06)	2.67 (0.08)	2.57 (0.33) / 0.02 (0.00)
FOS-ELM	OS-ELM	$M_{max} = 15$	8.26 (1.98)	8.90 (2.02)	9.46 (2.76)	10.52 (2.88)	11.76 (3.45)	9.78 (1.38)	1.55 (0.74) / 0.01 (0.00)
OAUE	OS-ELM	$M_{max} = 15$	1.94 (0.03)	2.11 (0.03)	2.21 (0.05)	2.29 (0.04)	2.40 (0.05)	2.19 (0.17)	9.76 (6.02) / 0.06 (0.04)
OB	OS-ELM	$M_{max} = 15$	2.68 (0.04)	2.51 (0.05)	2.67 (0.03)	2.61 (0.03)	2.55 (0.04)	2.61 (0.08)	3.13 (0.13) / 0.02 (0.00)
OEOA	OS-ELM	$M_{min} = 15$, $M_{max} = 15$	1.02 (0.02)	1.15 (0.02)	1.21 (0.03)	1.27 (0.02)	1.32 (0.02)	1.19 (0.12)	18.59 (1.14) / 0.11 (0.00)
OEOA	OS-ELM	$M_{min} = 5$, $M_{max} = 30$	**0.94 (0.04)**	**1.07 (0.03)**	**1.13 (0.02)**	**1.18 (0.03)**	**1.23 (0.02)**	**1.11 (0.11)**	34.88 (2.02) / 0.20 (0.01)
AddExp	λ_{DFF}OS-ELM	$M_{max} = 15$	0.63 (0.05)	0.65 (0.02)	0.67 (0.05)	**0.70 (0.05)**	**0.70 (0.05)**	0.67 (0.03)	5.27 (0.56) / 0.03 (0.00)
DOER	λ_{DFF}OS-ELM	$M_{max} = 15$	0.65 (0.02)	0.73 (0.02)	0.75 (0.02)	0.80 (0.02)	0.83 (0.02)	0.75 (0.07)	18.13 (2.20) / 0.11 (0.01)
EOS-ELM	λ_{DFF}OS-ELM	$M_{max} = 15$	0.67 (0.02)	0.69 (0.02)	0.77 (0.04)	0.77 (0.04)	0.77 (0.06)	0.73 (0.05)	2.32 (0.17) / 0.01 (0.00)
OAUE	λ_{DFF}OS-ELM	$M_{max} = 15$	0.59 (0.03)	**0.60 (0.03)**	**0.65 (0.05)**	0.74 (0.07)	0.85 (0.12)	0.69 (0.11)	9.47 (6.04) / 0.06 (0.04)
OB	λ_{DFF}OS-ELM	$M_{max} = 15$	0.88 (0.02)	0.89 (0.03)	0.99 (0.05)	0.98 (0.04)	0.97 (0.04)	0.94 (0.05)	2.77 (0.10) / 0.02 (0.00)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 15$, $M_{max} = 15$	0.65 (0.02)	0.73 (0.02)	0.76 (0.02)	0.80 (0.03)	0.83 (0.02)	0.75 (0.07)	18.08 (1.67) / 0.11 (0.01)
OEOA	λ_{DFF}OS-ELM	$M_{min} = 5$, $M_{max} = 30$	**0.53 (0.01)**	0.61 (0.02)	0.66 (0.03)	**0.70 (0.03)**	0.71 (0.03)	**0.64 (0.07)**	34.27 (2.13) / 0.20 (0.01)
Learn^{++}.NSE	ELM	$M_{max} = 15$	6.26 (1.95)	15.24 (11.48)	12.13 (4.44)	79.96 (98.39)	41.23 (24.70)	30.96 (30.49)	3.30 (0.97) / 0.02 (0.01)
OWE	ELM	$M_{max} = 15$	**2.93 (0.57)**	**3.08 (0.25)**	**3.09 (0.35)**	**4.08 (0.84)**	**4.65 (1.18)**	**3.57 (0.76)**	18.48 (4.31) / 0.11 (0.02)

[1]The MSE values have been multiplied by 10^3; [2]Av. and SD of the Processing Time on all samples (PTAS) and per sample (PTPS) over all values of T_0.

starts the system by creating an ensemble with one model; while OEOA starts the system by creating an initial pool of M_{max} models. The results indicate that OEOA without OA slightly outperforms DOER in most cases. However, in the polymerization reactor data set, with OS-ELM as base model, the average of the MSE over all values of T_0 for the DOER is 0.83×10^{-3}; while for the OEOA without OA it is 0.89×10^{-3}.

The results reveal that OEOA with OA is more time consuming when compared to the OEOA without OA. This is because the OA strategy requires more time to compute the best subset size. Additionally, the results show that sample-based ensembles with SW strategies (DOER, OEOA, OWE, and AddExp) require more processing time than batch-based ensembles (FOS-ELM, Learn^{++}.NSE, and OAUE). This is because, sample-based ensembles with SW train more models over time. Nevertheless, these ensembles outperform batch-based ensembles in prediction performance in most times. In the data set of large size (SRU; see Table 6), it is also observed that OEOA with OA requires significantly more processing time when compared to the other methodologies. This is because, more computations are necessary to select the best subset size and best models along time. Despite this, the processing time per sample (PTPS) is low (< 1 second). For example, in the data set of large size (SRU data set), using OEOA with OA and λ_{DFF}OS-ELM as the base model, the average of the PTPS is 0.20 seconds; while in the polymerization reactor data set (a small data set), using OEOA with OA and λ_{DFF}OS-ELM as the base model, the average of the PTPS is 0.12 seconds. Therefore, OEOA can be designed for practical use in real-world applications, reducing the time and maintenance costs of traditional measurement systems (e.g. laboratory measurement systems).

Recently, an ensemble of subset OS-ELM (ESOS-ELM) for class imbalance and concept drift in classification problems was proposed [50]. This contrasts with OEOA which is devoted to regression problems. ESOS-ELM comprises a *short-term memory*, representing the ensemble system; a *long-term memory*, representing an information storage module; and a change detection mechanism. The long-term memory stores models trained on old concepts that can be included to the ensemble when a recurring drift is detected. Results in [50] show that ESOS-ELM outperforms Learn^{++}.NSE and the Dynamic Weighted Majority (DWM) [51] (an approach similar to AddExp) in recurring drifts. OEOA does not use an information storage module as in ESOS-ELM. In OEOA, old models trained on old concepts are kept, and the best performing models on the current concept are used for prediction; however, old models that perform badly on the current concept may be replaced by new models over time. The main drawback in OEOA is that the previously acquired information may be excluded. Tests using the Friedman-GRA data set (see Table 3), a data set with recurring drifts, showed that OEOA outperforms Learn^{++}.NSE and AddExp. An important issue in ensemble development is the *diversity*. If the models provide the same output, there is nothing to be gained from their aggregation [31]. A drawback in ESOS-ELM is that the ensemble's models have the same number of hidden neurons, which may result in low diversity in the ensemble system. On the other hand, in the experiments performed with OEOA in this paper, for each model of the ensemble, the best number of hidden neurons is selected from an interval of $[1, 20]$, which increases the diversity of the ensemble.

6. Conclusions

A new on-line ensemble of regressor models using an OA method which is able to predict on-line variables in changing environments was proposed in this paper. The main contribution of the proposed ensemble is that it overcomes the problems of defining the optimal ensemble size and selecting of the set of most relevant models. These problems are solved by minimizing the ensemble's error on the newest sample. The results have shown that this strategy obtains better performance than combining all the models, in most cases. The proposed ensemble (OEOA) was shown to deliver more accurate estimations

of the output variables in industrial applications, as well as in several other cases, when compared to the other state-of-the-art ensembles in the literature. This paper also proposed the λ_{DFF}OS-ELM model, a new on-line ELM model using variable FF. λ_{DFF}OS-ELM was shown to have higher accuracy when compared to the OS-ELM; and it also improves the performance of well-known state-of-the-art ensembles. In general, OEOA and λ_{DFF}OS-ELM have high accuracy and fast adaptivity in non-recurring abrupt drifts (hyperplane data set), and in real-world applications. Thus, the proposed methods can be built for real industrial applications, reducing the time and maintenance costs of traditional measurement systems, such as laboratory measurement systems.

The proposed approaches have shown to have inferior performance in scenarios with local drifts. This may happen because the proposed method loses information about the old samples. Additionally, extra experiments may be necessary to tune the window' size in some applications. Future efforts can be devoted to use variables window' size that adapts according to the system characteristics. Moreover, as a future work, the authors would like to propose the dynamic selection of models using dynamic and fast meta-heuristics [52]; and a new λ_{DFF}OS-ELM algorithm that can be adapted both on sample and batch bases.

Acknowledgments

Symone Gomes Soares is supported by the Fundação para a Ciência e a Tecnologia (FCT) under the Grant SFRH/BD/68515/2010.

This work was supported by Project SCIAD "Self-Learning Industrial Control Systems Through Process Data" (reference: SCIAD/2011/21531) co-financed by QREN, in the framework of the "Mais Centro - Regional Operational Program of the Centro", and by the European Union through the European Regional Development Fund (ERDF).

The authors acknowledge the support of FCT project PEst-C/EEI/UI0048/2014. The authors are grateful to the Nanyang Technological University for sharing the implementation of the OS-ELM algorithm; and to Dr. Petr Kadlec, M. Eng. Ratko Grbić and Dr. Luigi Fortuna for sharing their data sets.

References

[1] L. Wang, Y. Zeng, T. Chen, Back propagation neural network with adaptive differential evolution algorithm for time series forecasting, Expert Systems with Applications 42 (2015) 855–863.

[2] S. M. Siniscalchi, T. Svendsen, C.-H. Lee, An artificial neural network approach to automatic speech processing, Neurocomputing 140 (2014) 326–338.

[3] S. Soares, R. Araújo, P. Sousa, F. Souza, Design and application of soft sensor using ensemble methods, in: Proc. of the 16th IEEE Int. Conf. on Emerging Technologies and Factory Automation, ETFA'11, 2011, pp. 1–8.

[4] Q. Dai, N. Liu, Alleviating the problem of local minima in backpropagation through competitive learning, Neurocomputing 94 (2012) 152–158.

[5] L. Sun, B. Chen, K.-A. Toh, Z. Lin, Sequential extreme learning machine incorporating survival error potential, Neurocomputing 155 (2015) 194–204.

[6] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Transactions on Neural Networks 17 (2006) 1411–1423.

[7] D. Wang, P. Wang, Y. Ji, An oscillation bound of the generalization performance of extreme learning machine and corresponding analysis, Neurocomputing 151, Part 2 (2015) 883–890.

[8] S. Haykin, Adaptive Filter Theory, 3rd ed., Prentice-Hall, Upper Saddle River, NJ, USA, 1996.

[9] W.-Y. Deng, Q.-H. Zheng, Z.-M. Wang, Cross-person activity recognition using reduced kernel extreme learning machine, Neural Networks 53 (2014) 1–7.

[10] X. Wang, M. Han, Online sequential extreme learning machine with kernels for nonstationary time series prediction, Neurocomputing 145 (2014) 90–97.

[11] T. Matias, D. Gabriel, F. Souza, R. Araújo, J. C. Pereira, Fault detection and replacement of a temperature sensor in a cement rotary kiln, in: Proc. of the 18th IEEE Int. Conf. on Emerging Technologies and Factory Automation, ETFA'13, 2013, pp. 1–8.

[12] J.-S. Lim, S. Lee, H.-S. Pang, Low complexity adaptive forgetting factor for online sequential extreme learning machine (OS-ELM) for application to nonstationary system estimations, Neural Computing and Applications 22 (2013) 569–576.

[13] L. Cao, H. Schwartz, A directional forgetting algorithm based on the decomposition of the information matrix, Automatica 36 (2000) 1725–1731.

[14] V. Bobál, J. Böhm, J. Fessl, J. Machácek, Digital Self-tuning Controllers: Algorithms, Implementation and Applications, Advanced Textbooks in Control and Signal Processing, Springer, 2005.

[15] R. Kulhavý, Restricted exponential forgetting in real-time identification, Automatica 23 (1987) 589–600.

[16] D. Wang, M. Alhamdoosh, Evolutionary extreme learning machine ensembles with size control, Neurocomputing 102 (2013) 98–110.

[17] Y. Lan, Y. C. Soh, G.-B. Huang, Ensemble of online sequential extreme learning machine, Neurocomputing 72 (2009) 3391–3395.

[18] P. Kadlec, B. Gabrys, Local learning-based adaptive soft sensor for catalyst activation prediction, AIChE Journal 57 (2011) 1288–1301.

[19] B. Lin, B. Recke, J. K. H. Knudsen, S. B. Jørgensen, A systematic approach for soft sensor development, Computers & Chemical Engineering 31 (2007) 419–425.

[20] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Transactions on Neural Networks 22 (2011) 1517–1531.

[21] R. Elwell, R. Polikar, Incremental learning in nonstationary environments with controlled forgetting, in: Proc. of the Int. Joint Conf. on Neural Networks, 2009, pp. 771–778.

[22] N. C. Oza, S. Russell, Experimental comparisons of online and batch versions of bagging and boosting, in: Proc. of the 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, KDD'01, 2001, pp. 359–364.

[23] J. Zhao, Z. Wang, D. S. Park, Online sequential extreme learning machine with forgetting mechanism, Neurocomputing 87 (2012) 79–89.

[24] H. Kaneko, K. Funatsu, Adaptive soft sensor based on online support vector regression and bayesian ensemble learning for various states in chemical plants, Chemometrics and Intelligent Laboratory Systems 137 (2014) 57–66.

[25] Y. Lv, J. Liu, T. Yang, D. Zeng, A novel least squares support vector machine ensemble model for NOx emission prediction of a coal-fired boiler, Energy 55 (2013) 319–329.

[26] J. Z. Kolter, M. A. Maloof, Using additive expert ensembles to cope with concept drift, in: Proc. of the 22nd Int. Conf. on Machine Learning, ACM, 2005, pp. 449–456.

[27] S. G. Soares, R. Araújo, An on-line weighted ensemble of regressor models to handle concept drifts, Engineering Applications of Artificial Intelligence 37 (2015) 392–406.

[28] S. G. Soares, R. Araújo, A dynamic and on-line ensemble regression for changing environments, Expert Systems with Applications 42 (2015) 2935–2948.

[29] D. Brzezinski, J. Stefanowski, Combining block-based and online meth-

ods in learning ensembles from concept drifting data streams, Information Sciences 265 (2014) 50–67.
[30] Z.-H. Zhou, J. Wu, W. Tang, Ensembling neural networks: Many could be better than all, Artificial Intelligence 137 (2002) 239–263. Code available at `http://lamda.nju.edu.cn/files/Gasen.zip`.
[31] S. Soares, C. H. Antunes, R. Araújo, Comparison of a genetic algorithm and simulated annealing for automatic neural network ensemble development, Neurocomputing 121 (2013) 498–511.
[32] I. Partalas, G. Tsoumakas, E. V. Hatzikos, I. Vlahavas, Greedy regression ensemble selection: Theory and an application to water quality prediction, Information Sciences 178 (2008) 3867–3879.
[33] G. Coelho, F. Von Zuben, The influence of the pool of candidates on the performance of selection and combination techniques in ensembles, in: Proc. of the Int. Joint Conf. on Neural Networks, IJCNN'06, 2006, pp. 5132–5139.
[34] A. Lazarevic, Z. Obradovic, Effective pruning of neural network classifier ensembles, in: Proc. of the Int. Joint Conf. on Neural Networks, volume 2 of *IJCNN'01*, 2001, pp. 796–801.
[35] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (2006) 489–501.
[36] A. Ben-Israel, T. N. Greville, Generalized Inverses: Theory and Applications, 2nd ed., Springer-Verlag, New York, USA, 2003.
[37] C. R. Rao, S. K. Mitra, Generalized inverse of a matrix and its applications, in: Proc. 6th Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics, University of California Press, Berkeley, CA, USA, 1972, pp. 601–620. URL: `http://projecteuclid.org/euclid.bsmsp/1200514113`.
[38] P. Maponi, The solution of linear systems by using the sherman-morrison formula, Linear Algebra and Its Applications 420 (2007) 276–294.
[39] J. Mendes, R. Araújo, F. Souza, Adaptive fuzzy identification and predictive control for industrial processes, Expert Systems with Applications 40 (2013) 6964–6975.
[40] V. Bobál, P. Chalupa, Self-Tuning Controllers Simulink Library, Zlín, Czech Republic, 2008. `http://www.mathworks.com/matlabcentral/fileexchange/8381-stcsl-standard-version`.
[41] R. Kulhavý, Probabilistic Identification of Time-Variable Systems with Unknown Model of Parameter Evolution, PhD thesis, Institute of Information Theory and Automation of Czechoslovak Academy of Sciences, Praha, Czechoslovakia, 1985. (in Czech).
[42] R. Klinkenberg, Meta-learning, model selection, and example selection in machine learning domains with concept drift, in: Proc. of Annual Workshop of the Special Interest Group on Machine Learning, Knowledge Discovery, and Data Mining, FGML-2005, 2005, pp. 164–171.
[43] E. M. D. Santos, R. Sabourin, P. Maupin, Overfitting cautious selection of classifier ensembles with genetic algorithms, Information Fusion 10 (2009) 150–162.
[44] E. Ikonomovska, Algorithms for Learning Regression Trees and Ensembles on Evolving Data Streams, PhD's thesis, Jožef Stefan Int. Postgraduate School, 2012.
[45] J. H. Friedman, Multivariate adaptive regression splines, The Annals of Statistics 19 (1991) 1–67.
[46] R. Grbić, D. Slišković, P. Kadlec, Adaptive soft sensor for online prediction and process monitoring based on a mixture of gaussian process models, Computers & Chemical Engineering 58 (2013) 84–97.
[47] L. Fortuna, S. Graziani, A. Rizzo, M. G. Xibilia, Soft Sensors for Monitoring and Control of Industrial Processes (Advances in Industrial Control), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
[48] D. L. Shrestha, D. P. Solomatine, Experiments with AdaBoost.RT, an improved boosting scheme for regression, Neural Computation 18 (2006) 1678–1710.
[49] H. J. Galicia, Q. P. He, J. Wang, Comparison of the performance of a reduced-order dynamic PLS soft sensor with different updating schemes for digester control, Control Engineering Practice 20 (2012) 747–760.
[50] B. Mirza, Z. Lin, N. Liu, Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift, Neurocomputing 149, Part A (2015) 316–329.
[51] J. Z. Kolter, M. A. Maloof, Dynamic weighted majority: An ensemble method for drifting concepts, Journal of Machine Learning Research 8 (2007) 2755–2790.
[52] V. Mangat, R. Vig, Novel associative classifier based on dynamic adaptive PSO: Application to determining candidates for thoracic surgery, Expert Systems with Applications 41 (2014) 8234–8244.

Symone G. Soares received her B.Sc. degree in Computer Engineering from the Pontifical Catholic University of Goiás (PUC-GO), Brazil, in 2009. Actually she is pursuing her Ph.D. degree in Electrical and Computer Engineering at the University of Coimbra, Portugal. Her research interests include machine learning, optimization techniques, computational intelligence modeling, and Soft Sensors for industrial applications.

Rui Araújo received the B.Sc. degree ("Licenciatura") in Electrical Engineering, the M.Sc. degree in Systems and Automation, and the Ph.D. degree in Electrical Engineering from the University of Coimbra, Portugal, in 1991, 1994, and 2000 respectively. He joined the Department of Electrical and Computer Engineering of the University of Coimbra where he is currently an Assistant Professor. He is a founding member of the Portuguese Institute for Systems and Robotics (ISR-Coimbra), where he is now a researcher. His research interests include computational intelligence, intelligent control, computational learning, fuzzy systems, neural networks, estimation, control, robotics, mobile robotics and intelligent vehicles, robot manipulators control, sensing, soft sensors, automation, industrial systems, embedded systems, real-time systems, and in general architectures and systems for controlling robot manipulators, mobile robots, intelligent vehicles, and industrial systems.